UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS
Departamento de Estadística e Investigación Operativa

TESIS DOCTORAL

Contribuciones al análisis de riesgos adversarios

(Contributions to Adversarial Risk Analysis)

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Jorge González Ortega

Director

David Ríos Insua

Madrid
Ed. electrónica 2019

UNIVERSIDAD COMPLUTENSE DE MADRID
UNIVERSIDAD POLITÉCNICA DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS (UCM)
Departamento de Estadística e Investigación Operativa

DOCTORADO EN
Ingeniería Matemática, Estadística e Investigación Operativa (IMEIO)

TESIS DOCTORAL

Contribuciones al Análisis de Riesgos Adversarios

(Contributions to Adversarial Risk Analysis)

MEMORIA PRESENTADA POR
Jorge González Ortega

DIRECTOR
David Ríos Insua

Madrid, 2018

Las preguntas que no podemos contestar son las que más nos enseñan. Nos enseñan a pensar. Si le das a alguien una respuesta, lo único que obtiene es cierta información. Pero si le das una pregunta, él buscará sus propias respuestas.

El Temor de un Hombre Sabio
Patrick Rothfuss

Agradecimientos

El período predoctoral de tres años que me ha llevado a la redacción de esta tesis doctoral ha sido frenético y, a la vez, enormemente enriquecedor. Durante el mismo, como en todo camino, ha habido numerosos altibajos en los que me han ayudado el apoyo, la comprensión y los conocimientos de quienes me han acompañado. Sea para ellos este pequeño espacio.

No puedo sino comenzar por agradecerle a David toda la confianza que ha depositado en mí desde el primer día y su paciencia frente a mi perfeccionismo e idealismo. Es un ejemplo de esfuerzo y trabajo constantes además de una fuente de inspiración, no sólo en el ámbito académico, sino en el personal exprimiendo las 24 horas del día al máximo sin perder el buen humor. Todo lo que me ha enseñado, las oportunidades que me ha brindado y su preocupación por mi bienestar hacen de él más que un director, un amigo.

El segundo lugar está reservado, sin duda alguna, para mis padres Yolanda y Luis. Podría decir que siendo ambos matemáticos de formación mi destino estaba ya escrito, pero a lo largo de toda mi vida siempre me han dado su apoyo incondicional y la libertad de decidir qué hacer, proporcionándome valiosos consejos y facilitándome todos los recursos a su alcance. Nada de lo que aquí pueda escribir reflejará fielmente lo agradecido que me siento por toda una vida de tiempo y cuidados. Gracias, os quiero.

Mención especial merece también mi hermano Dani que, pese a la distancia que nos separa, ha sabido sacar el tiempo necesario para animarme e interesarse por mis avances en largas conversaciones vía Skype. Hago extensible esta gratitud al resto de mi familia, pues igualmente han sabido alentarme para que llevara esta empresa a buen puerto.

Por supuesto, no puedo olvidarme de todos esos amigos que han estado a mi lado para seguir demostrando que en esta vida no sólo es importante el trabajo. Gracias a Viki por estar siempre ahí; a Rober, Luis M., Enrique, Julia, Paloma, Jesús y Luis I. por tantos años de amistad desde que nos conociéramos estudiando en la facultad; a Celestino, Jorge, Iván y familia por ponerle ritmo (cubano) a mi vida; a Edu y Andrés a quienes he tenido la suerte de conocer más recientemente; y a Alma por

recordarme cómo disfrutar del viaje.

También a Arya, la mejor compañera de piso que nadie podría desear. Por su aguante ante mis largas horas delante del ordenador pese a sus ganas de jugar y sus cariños y ronroneos que me han empujado a seguir adelante incluso en los días más complicados. Y a Sanju, por traerla a mi vida y todo lo que aprendimos juntos.

Tres años son mucho tiempo y puedo afirmar que me he sentido muy cómodo trabajando tanto en el ICMAT como en la UCM. De ello tienen el mérito mis compañeros del grupo SPOR, tanto los doctorandos April, Roi, Simón, Víctor, Alberto y Aitor, como los investigadores David GU., Alberto T., Pablo A. y Pablo S. Así mismo, quiero destacar a Antonio que, siendo miembro de dicho equipo, ha tenido la responsabilidad adicional de ejercer como mi tutor en el programa de doctorado estando constantemente disponible para resolver todas mis dudas y acometer esas tareas burocráticas más ingratas. Además quiero reivindicar la labor del personal de administración y servicios del ICMAT y de la cafetería de la UCM, demasiados como para nombrarlos a todos, siempre amables y haciéndome sentir como en casa.

He tenido la fortuna de disfrutar de una estancia predoctoral en Washington, DC. Allí fui recibido con los brazos abiertos en la Universidad George Washington por Refik, pero también por el resto del departamento y, en especial, por los doctorandos Didem, Atilla y Kian. No puedo omitir tampoco a Mark, con quien conviví todo ese tiempo y pude trabar una maravillosa amistad que no entiende de fronteras ni cambios horarios.

Para concluir, quiero agradecer al Ministerio de Economía y Competitividad de España la financiación a través del contrato predoctoral FPI Severo Ochoa BES-2015-072892 ligado al grupo de investigación SPOR del ICMAT. Además, por su financiación indirecta, quiero también reconocer la ayuda recibida por el Ministerio de Economía y Competitividad de España mediante el programa "Severo Ochoa" para Centros de Excelencia en I+D (SEV-2015-0554) y el proyecto MTM2015-72907-EXP; el Ministerio de Economía e Innovación de España al amparo del programa MTM2014-56949-C3-1-R; el Programa Internacional de Cooperación Europea en el Campo de la Investigación Científica y Técnica (COST) con la Acción IS1304 en Juicios Expertos; y el Programa para la Investigación y la Innovación en la Unión Europea H2020 bajo el acuerdo no. 740920 (CYBECO).

Índice

Listas

Algoritmos

Figuras

Tablas

Resumen

La teoría de juegos y otros paradigmas de toma de decisiones en grupo, véase Gibbons [1992], se han considerado durante mucho tiempo inadecuados en la mayoría de aplicaciones al análisis de riesgos. Sin embargo, gracias al apremio por tener en cuenta las estrategias de los adversarios en contextos como la política antiterrorista, la competencia económica o la ciberseguridad, ha surgido un creciente interés en estas metodologías otrora descartadas.

El Análisis de Riesgos Adversarios (ARA) [Ríos Insua et al., 2009] es un marco emergente para el apoyo a un decisor que se enfrenta a oponentes en situaciones de conflicto cuyas consecuencias son aleatorias y determinadas por las acciones de todos los agentes involucrados. El ARA brinda ayuda prescriptiva y unilateral al decisor, maximizando su utilidad esperada mientras las decisiones de sus adversarios se tratan como variables aleatorias. Para encontrar su mejor plan de acción, el ARA modeliza y resuelve los problemas de toma de decisiones de los oponentes, bajo suposiciones sobre su racionalidad, introduciendo distribuciones de probabilidad subjetivas en todas las cantidades inciertas y, de esta forma, encontrando una predicción sobre cada una de sus elecciones. A veces, la estimación de dichas distribuciones de probabilidad puede conducir a una jerarquía de problemas de toma de decisiones anidados, como se describe en Ríos y Ríos Insua [2012], de forma vinculada al concepto de razonamiento de nivel k [Stahl y Wilson, 1995].

En contraste con los enfoques de teoría de juegos habituales, el ARA no asume las estándar, pero poco realistas al menos en aplicaciones de seguridad, hipótesis de conocimiento común, criticadas por ejemplo en Hargreaves-Heap y Varoufakis [1995] o Raiffa et al. [2002], según las cuales los agentes comparten información sobre sus creencias sobre las utilidades y probabilidades de cada uno. Así mismo, amplía el espectro limitado de los conceptos de solución y racionalidades ligados a los equilibrios de Nash y nociones derivadas comprendidas dentro del paradigma actual de la teoría de juegos (no-cooperativos).

El objeto de esta tesis doctoral, titulada *Contribuciones al Análisis de Riesgos Adversarios*, es proporcionar nuevos modelos en el contexto del ARA. A fin de lograr esto, se cubren tres objetivos principales: (O1) el desarrollo de métodos de representación en ARA; (O2) la mejora de la modelización del razonamiento estratégico;

y (O3) la implementación de modelos ARA.

Los resultados comprenden un enfoque algorítmico ARA para evaluar diagramas de influencia bi-agente propios (Objetivo O1), en el cual se apoya al decisor empleando una estrategia de razonamiento de nivel 2 y se consideran interacciones secuenciales y simultáneas generales; una perspectiva ARA sobre cómo gestionar la incertidumbre de concepto (Objetivo O2), incluyendo los fundamentos de una teoría de decisión estadística adversaria que engloba un amplio conjunto de problemas estadísticos relevantes; y una solución ARA para el problema de contraste de hipótesis con adversarios (Objetivo O3), extendido a un modelo de aceptación de lotes con observaciones parciales en oposición a la premisa de información completa en la estructura básica.

A partir de los contenidos de esta tesis doctoral se han elaborado los siguientes cuatro artículos en diversos estados de publicación:

González-Ortega, J.; Ríos Insua, D. & Cano, J. (2018). Adversarial risk analysis for bi-agent influence diagrams: An algorithmic approach. *European Journal of Operational Research*, en prensa, `https://doi.org/10.1016/j.ejor.2018.09.015`.

González-Ortega, J.; Ríos Insua, D.; Ruggeri, F. & Soyer, R. (esp. 2019). Hypothesis testing in presence of adversaries. Enviado a *The American Statistician*.

Ríos Insua, D.; Banks, D.L.; Ríos, J. & González-Ortega, J. (2017). Adversarial risk analysis. En *Wiley StatsRef: Statistics Reference Online*, `https://doi.org/10.1002/9781118445112.stat07972`.

Ríos Insua, D.; González-Ortega, J.; Banks, D.L. & Ríos, J. (2018). Concept uncertainty in adversarial statistical decision theory. En *The Mathematics of the Uncertain: A Tribute to Pedro Gil*, 2018 ed., Springer, Cham, Suiza, 527–542.

Abstract

Game theory and other group decision-making paradigms, see Gibbons [1992], have been regarded for a long time inadequate in most risk analysis applications. However, thanks to the urge to take into account the strategies of adversaries in contexts such as counterterrorism, economic competition or cybersecurity, a growing interest on these once dismissed methodologies has arisen.

Adversarial Risk Analysis (ARA) [Ríos Insua et al., 2009] is an emergent framework for supporting a decision-maker (she) who faces opponents (he/they) in conflict situations for which the consequences are random and determined by all interacting agents' actions. ARA provides one-sided prescriptive help to the decision-maker, maximising her expected utility while treating her adversaries' decisions as random variables. To find her best course of action, ARA models and solves the opponents' own decision-making problems, under assumptions about their rationality, introducing subjective probability distributions on all uncertain quantities and, thus, finding a forecast over each of their choices. Sometimes the assessment of such probability distributions may lead to a hierarchy of nested decision-making problems, as described in Ríos and Ríos Insua [2012], which is related to the concept of level-k thinking [Stahl and Wilson, 1995].

In contrast to usual game-theoretic approaches, ARA does not assume the standard, but unrealistic at least in security applications, common knowledge assumptions, criticised in e.g. Hargreaves-Heap and Varoufakis [1995] or Raiffa et al. [2002], according to which the agents share information about their beliefs of each other's utilities and probabilities. Besides, it broadens the limited scope of solution concepts and rationalities associated with Nash equilibria and derived notions within the current paradigm of (non-cooperative) game theory.

The aim of this PhD thesis, titled *Contributions to Adversarial Risk Analysis*, is to provide new models in the context of ARA. In order to achieve this, three main objectives are covered: (O1) the development of representation methods in ARA; (O2) the enhancement of the modelling of strategic reasoning; and (O3) the implementation of ARA models.

The results comprise an ARA algorithmic approach to evaluate proper bi-agent influ-

ence diagrams (Objective O1), in which the decision-maker is supported employing a level-2 thinking strategy and general sequential and simultaneous interactions are considered; an ARA perspective on how to handle concept uncertainty (Objective O2), including the foundations of an adversarial statistical decision theory that encompasses an extensive set of relevant statistical problems; and an ARA solution to the adversarial hypothesis testing problem (Objective O3), extended to a batch acceptance model with partial observations as opposed to the assumption of complete information in the basic structure.

From the contents of this PhD thesis the following four papers have been elaborated under diverse publishing states:

González-Ortega, J.; Ríos Insua, D. & Cano, J. (2018). Adversarial risk analysis for bi-agent influence diagrams: An algorithmic approach. *European Journal of Operational Research*, in press, `https://doi.org/10.1016/j.ejor.2018.09.015`.

González-Ortega, J.; Ríos Insua, D.; Ruggeri, F. & Soyer, R. (exp. 2019). Hypothesis testing in presence of adversaries. Submitted to *The American Statistician*.

Ríos Insua, D.; Banks, D.L.; Ríos, J. & González-Ortega, J. (2017). Adversarial risk analysis. In *Wiley StatsRef: Statistics Reference Online*, `https://doi.org/10.1002/9781118445112.stat07972`.

Ríos Insua, D.; González-Ortega, J.; Banks, D.L. & Ríos, J. (2018). Concept uncertainty in adversarial statistical decision theory. In *The Mathematics of the Uncertain: A Tribute to Pedro Gil*, 2018 ed., Springer, Cham, Switzerland, 527–542.

Chapter 1

Introduction

We live in a competitive world. Moreover, in a competitive positioning world where everything is relative and our success entails someone's defeat. From a young age we are taught that there are winners and there are losers; we are taught there are 1st, 2nd and 3rd places in competitions. We are encouraged to compete with others and outscore everyone else.

Yet, this is not just a contemporary attitude. Chapter 4 of *On the Origin of Species* [Darwin, 1859] already mentions "natural selection, or the survival of the fittest". That is, the better adapted for the immediate, local environment individuals are, the more likely they are to survive against the competitive nature of the world.

Accordingly, our day to day life provides many competitive situations we have to constantly deal with: our professional career choices; a wide variety of retailers offering all sorts of products, qualities and prices; different social media struggling to draw our attention... Furthermore, security, politics, environmental regulation and disease control are all examples of areas in which multiple agents with distinct goals collide while pursuing their own interests.

These may lead to cooperative or non-cooperative circumstances. Within cooperative conditions, groups of agents ("coalitions") that compete between them are formed due to the existence of external enforcement of cooperative behaviour. In opposition, a non-cooperative context arises when there is either no possibility to forge alliances or all agreements are self-enforcing, that is, they stand as long as the counterparts believe the agreement is mutually beneficial and is not breached by either party.

Recent events such as terrorist attacks or economic crises have stressed the need for decision-makers to take into account the strategies of their opponents in conflict situations. The classic approach in mathematics and economics to such problems has been (non-cooperative) *game theory* [von Neumann and Morgenstern, 1944].

1.1 Game theory

For decades, game theory and other group decision-making paradigms, see Gibbons [1992], have been considered of little use in practical risk management problems. However, this point of view has recently become less preponderating because there has been: (i) a mathematical sophistication of key business sectors which is now being used to define adversary-aware strategies for competitive decisions in marketing, auctions and other areas; (ii) an increase in regulatory legislation to balance, in a transparent and credible way, competing interests such as safety, growth and environmental impact; (iii) a substantial demand of public investment in protective responses to high-profile terrorist attacks with a major concern on their effectiveness; and (iv) an acquaintance of random and potentially large financial penalties for myopic protection in cybersecurity.

Game theory is a rich discipline which has been, and still is, thoroughly studied, but is not adequate for the complexity of many real-world applications. A main drawback of this methodology is its underlying common knowledge assumptions, criticised in e.g. Hargreaves-Heap and Varoufakis [1995] or Raiffa et al. [2002]. Most versions of non-cooperative game theory assume that adversaries know not only their own payoffs, preferences, beliefs and possible actions, but also those of their opponents. When there is uncertainty in the game, it is typically assumed that players have common probabilities or, at least, a joint probability distribution over their types known to all of them, as in games of incomplete information (Bayesian games) [Harsanyi, 1967]. These common knowledge assumptions allow for a symmetric joint normative analysis in which players maximise their expected utilities, and expect other players to do the same. Their decisions can then be anticipated and predicted through Bayes-Nash equilibria and related refinements. However, in contexts such as counterterrorism or cybersecurity, players will not generally have so much knowledge about their opponents.

The other main objection derives from the fact that classical game theory assumes *perfect rationality*, so that players are regarded as remarkably intelligent and capable of framing and solving extremely complicated rational arguments to come up with the most adequate solution. On one hand, it is quite obvious that nobody would play chess in the same way against a ten year old kid as against Magnus Carlsen, but rather play with respect to the opponent's skills and game style. The amount of effort and acceptable risk depends upon how strong, competent and experienced the adversary seems to be. On the other hand, human beings have only a limited amount of time, knowledge, memory and resources available when making rational choices, which calls for *bounded rationality* [Gigerenzer and Selten, 2000] instead. This has mainly been incorporated to game theory, as reviewed in Conlisk [1996] for economic theory, through satisficing an optimising procedures like suboptimization, heuristics, evolutionary approaches or deliberation, and also by means of hierarchical thinking theories such as level-k thinking [Stahl and Wilson, 1995] and cognitive hierarchy

Camerer et al. [2004]. However, all these methods still rely to some extent on common knowledge assumptions and relate to debatable equilibria solution concepts.

1.2 Decision analysis

An alternative to game theory for dealing with conflict situations may be found in *decision analysis*. In the context of strategic opponents, it was simultaneously proposed by Raiffa [1982] and Kadane and Larkey [1982]. Developed from Bayesian statistics, it presumes that decision-makers seek to maximise their expected utility or, equivalently, minimise their expected loss. This results in a fundamentally different solution concept to that of the Nash equilibrium prevalent in game theory, which assumes that each player knows the equilibrium strategies of the other players and no player has anything to gain by independently changing their own strategy.

Decision analysis is Bayesian in that it requires decision-makers to assess probability distributions over their opponents' actions. In most settings, the frequentist definition of probability is untenable, as there is no prospect of repeated independent trials. Hence, decision-analysts typically use personal probabilities, in the Savage [1954] sense, reflecting their subjective beliefs about the likely actions of the other agents.

However, decision analysis in conflict situations is not exempt of controversy. It was deemed contrary to the spirit of game theory by Harsanyi [1982], arguing that probability assessment of the adversaries' actions ought to be based upon an analysis of their rational behaviour, and not upon their actual behaviour in similar games nor upon judgements about their knowledge, motives or strategic capacity. In addition, Myerson [1991] challenged the difficulty derived from trying to determine the subjective probability distribution over the other players' strategies devising how they conceive our own strategy and adapt to it, which could be the preamble of a potentially infinite recursive problem in which we think what they would think that we think they think...

In decision analysis, the problem is considered from the standpoint of a single agent (she), solely using her beliefs and knowledge, rather than trying to provide a solution to all agents' problems simultaneously. The supported decision-maker is assumed to have: (i) subjective conditional probabilities over the outcome(s) for every set of possible choices; (ii) perfect knowledge of her own utility function; and (iii) a subjective probability over the actions of each opponent. These subjective beliefs held by the agent regarding the moves of her adversaries (he/they) pose the major difficulty in implementing a decision analysis. When credible intelligence about the opponents is available, it becomes an extremely valuable input. However, this is often either unavailable or insufficient, so some kind of game-theoretic reasoning

might be convenient to model the decision-maker's uncertainty about the other agents' actions.

Traditional decision analysis has not explicitly addressed how to use a strategic perspective to construct one's personal probabilities. Though Kadane [2009] advises Bayesians to express their uncertainty on other players' choices through probabilities, he gives no prescription for how to arise them from a combination of strategic analysis of the game and whatever knowledge they may have of their opponents. To fill that gap, *adversarial risk analysis* has been conceived.

1.3 Adversarial risk analysis

Among all methodologies for conflict situations comprehended under the scope of the decision analysis field, this PhD thesis focuses on the so called Adversarial Risk Analysis (ARA) approach [Ríos Insua et al., 2009]. The ARA framework stems from observations in Parnell et al. [2007] and is founded on the idea that the analyst should build a model for her adversaries' decision-making processes. Within those models, she would solve the problem from the perspective of each opponent, introducing subjective probability distributions on all unknown quantities, so as to obtain a forecast over the actions of each of them. Those distributions would then enable the analyst to maximise her expected utility.

There are many ways in which these models can be developed, depending on how the adversaries' thinking is devised. Again, in the counterterrorism context, the decision-maker might study historical terrorist attempts and assume that the terrorists' probabilities and utilities are related to those of their predecessors. Alternatively, she may analyse the terrorists' previous public statements to assess their utility from successful attacks and infer their chances of success. If she believes they are irrational, she might base the model on random behaviour treating them much as natural hazards; whilst if she considers them rational, she could presume they will maximise their expected utility or, in case of being extremely risk averse, minimise their worst-case outcome. She may even consider that they resort to ARA too, mirroring her analysis. In any case, the decision-maker could assign a subjective probability to each option, expressing her uncertainty about the kind of terrorists she faces, and use a mixture model to describe their reasoning.

ARA provides a way forward to game theory with respect to common knowledge assumptions, as they are no longer required, with just one decision-maker being supported against all other agents employing her own perspective and subjective beliefs. However, the distinguishing feature of ARA is that it emphasises the advantage of building a model for the strategic reasoning of an opponent. Essentially, ARA is an attempt to combine both strategic reasoning about adversaries and probabilistic

treatment of aleatory outcomes. It is an emerging perspective that has attractive features in counterterrorism issues, as described in Ríos Insua et al. [2009] and Merrick and Parnell [2011], though it has many more potential applications. Additional insights on the combination of risk analysis and game theory can be found in e.g. Cox Jr. [2009].

1.4 State of the art

Research in ARA has mainly focused on the resolution of specific problems through which general models have been extrapolated. In this way, relatively simple ARA models with basic sequences of defence and attack movements have been developed. This PhD thesis covers two main lines of research. The first one is to continue with the actual research strategy in ARA, contributing to the development of a comprehensive and generic ARA framework which may be applied to specific problems in different areas such as counterterrorism, cybersecurity or adversarial classification. The second one explores diverse rationality paradigms that may be used for adversaries and how to model them.

In this section, preceding relevant results in both lines of research will be covered to lay the foundation of the work in this PhD thesis. Specific open questions are raised and the significance of the ARA framework is highlighted.

1.4.1 Representation methods

As mentioned in Section 1.3, the ARA approach towards non-cooperative games is based on supporting a single agent in her decision-making problem by building a model for each of her opponents' strategies, incorporating uncertainty through probability distributions. To do so, the game-theoretic structure of the problem must be exploited and different representation methods are found in the literature.

Standard game representations, both the *normal matrix form* and the *extensive game tree form* [Luce and Raiffa, 1957], are perhaps the most frequently used and well-known methods. They succeed in dealing with simple problems, yet grow exponentially with the number of variables and cannot cope with complex models. Furthermore, they conceal the underlying structure of chance and decision variables which obscures both the decision-making process and communication. Therefore, they are inadequate to an ARA approach.

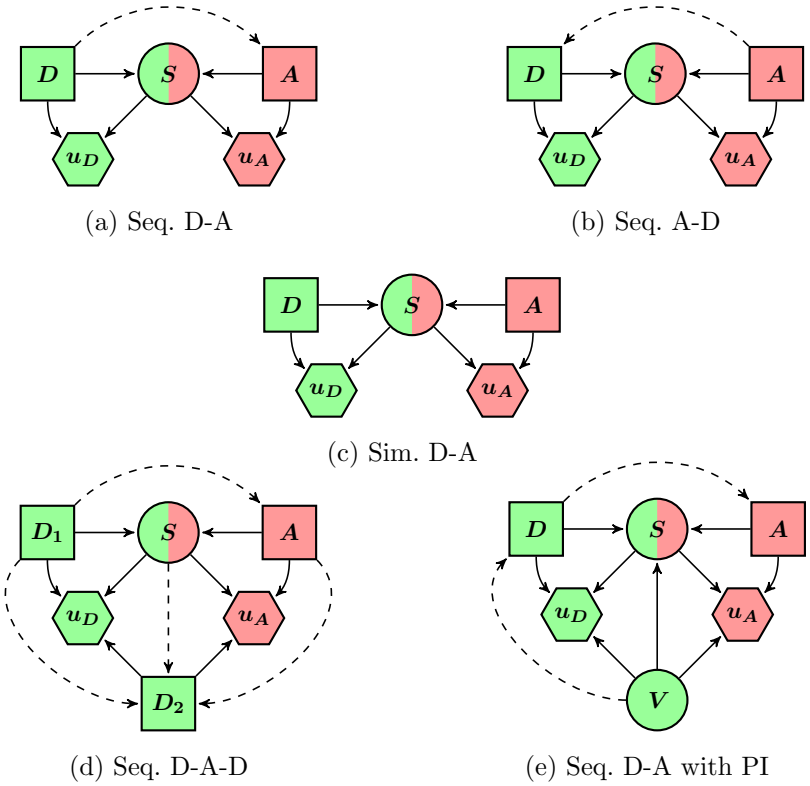
Probabilistic graphical models [Pearl, 1988] solve this problem using a directed graph structure, where the nodes represent variables and the edges the direct dependence of one variable on another, making explicit the conditional independence properties of

the variables. These graphs are called *probabilistic* or *Bayesian networks* and have clear and formal semantics that define them as a representation of a probability distribution over the state space specified by the variables.

A natural extension of Bayesian networks to the decision-theoretic framework are *Influence Diagrams* (IDs) [Miller et al., 1976]. In addition to chance variables, IDs include decision variables, whose value the decision-maker chooses as part of her strategy, and utility variables, which evaluate the expected utility of the resulting outcomes. In his landmark paper, Shachter [1986] developed an algorithm capable of evaluating any proper ID, determining the optimal policy for its decisions, and proposed extending the computation of optimal decision policies in IDs to the multi-agent case as a critically important problem. So far, this suggestion has been confronted from a (non-cooperative) game-theoretic perspective through problem decomposition strategies relying on decision independence concepts, e.g. employing conditional independence [Smith, 1996] or strategic independence [La Mura, 2000], but specially stemming from Koller and Milch [2003] who introduced *Multi-Agent Influence Diagrams* (MAIDs) and provided algorithms for finding Nash equilibria in problems modelled as such using the notions of strategic relevance and s-reachability. The problem has not been fully assessed from the ARA viewpoint.

Specific cases dealing with protection from intelligent threats such as anti-IED defence in routing problems [Wang and Banks, 2011] or preventing ships from piracy risks [Sevillano et al., 2012] have been modelled and solved with an ARA approach. Banks et al. [2015] provide a broad review of applications in a variety of contexts. These and other applications have been dealt with relatively simple ARA models with basic sequences of attack and defence movements. Indeed, a number of templates which may be viewed as basic building blocks for general security risk analysis problems can be identified, as in Brown et al. [2006], Zhuang and Bier [2007], Brown et al. [2008] or Hausken [2011]. They differ in the way and order in which attack and defence actions take place within the global sequence of decisions and events, as well as in the information revealed. Figure 1.1 presents five of these basic templates, with self-explanatory names: (a) the Sequential Defend-Attack (Seq. D-A) model; (b) the Sequential Attack-Defend (Seq. A-D) model; (c) the Simultaneous Defend-Attack (Sim. D-A) model; (d) the Sequential Defend-Attack-Defend (Seq. D-A-D) model; and, finally, (e) the Sequential Defend-Attack with Private Information (Seq. D-A with PI) model. They are covered in full detail in Ríos and Ríos Insua [2012] and Ríos Insua et al. [2013].

Beyond these templates, general adversarial problems between two agents in which complex interactions are allowed, typically consisting of intermingled sequential and simultaneous movements spanning across the relevant planning period, need yet to be considered. This PhD thesis provides an algorithmic approach to solve (evaluate) general bi-agent adversarial problems from an ARA perspective using Bi-Agent Influence Diagrams (BAIDs).



(a) Seq. D-A (b) Seq. A-D

(c) Sim. D-A

(d) Seq. D-A-D (e) Seq. D-A with PI

Figure 1.1: BAIDs for five template models

1.4.2 Rationality paradigms

The key aspect of ARA, in contrast to the standard game-theoretic approach, is its usage of a model for the strategic reasoning of the opponents, that is, the adversarial component. Understanding and judging the motives, skills and way of thinking of the adversaries makes a difference in the assessment of risks related to the decision-making process.

Frequently, in decision-making contexts, it is typical to address just two types of uncertain phenomena, see e.g. Parry [1996], Walker et al. [2003] or Refsgaard et al. [2007], which could be described as follows in adversarial problems:

- *Aleatory uncertainty.* It refers to the randomness of the outcomes that the agents receive, thus conditional on their choices.
- *Epistemic uncertainty.* It involves the strategic choices of intelligent adversaries, as driven by unknown preferences, beliefs and capabilities.

However, as mentioned in Section 1.1, a renewed interest on non-cooperative group

decision-making paradigms that effectively take into account the strategies of adversaries has arisen of late. This has entailed the need to consider a third type of uncertainty:

- *Concept uncertainty.* It addresses how opponents frame the problem in relation to questions like what kind of strategic analysis do they use, whether they are rational or how deeply do they think. Often, the solution used will determine the epistemic uncertainties that are relevant and, consequently, the aleatory uncertainties.

Most authors in this context would focus on game-theoretic methods based on variants of Nash equilibria, see e.g. Gibbons [1992]. However, the ARA framework allows to weaken the Nash equilibria common knowledge assumptions and provide more flexible models for opponent behaviour. In a comparison of methods for adversarial risk management, Merrick and Parnell [2011] preferred ARA precisely because it handles and apportions these separate uncertainties more explicitly. In this respect, ARA may be viewed as a *Fermitisation* strategy, as defined by Tetlock and Gardner [2015], since it simplifies the assessment of complex uncertainties by considering decompositions of complex problems into simpler ones in which the assessment is easier.

ARA solutions are usually developed in the context of four common rationality paradigms: *non-strategic thinking*, *Nash equilibrium*, *level-k thinking* and *mirroring argument*. A detailed description of each of them may be found in Banks et al. [2015]. Research has been done to manage the uncertainty about the strategic reasoning used by the adversary (mixture models) and measure their validity (Bayesian updating), e.g. Ríos Insua, Banks and Ríos [2016], yet further work remains to be done as only simple two-agent situations have been considered. This PhD thesis develops more complex models and test them to help gain a deeper insight into how to conduct concept uncertainty.

1.4.3 Adversarial applications

A main motivation for ARA developments arises from security and counterterrorism contexts, but potential applications abound in many other areas: from competitive marketing to disease control going through auctions, cybersecurity, social robotics or supply chain management. A relevant issue across many of them is that of *Adversarial Hypothesis Testing* (AHT).

Hypothesis testing is one of the fundamental problems in statistical inference [French and Ríos Insua, 2000]. Though subject to debate, Berger and Sellke [1987], Berger [2003], Johnson [2013] or Wasserstein and Lazar [2016], it has been thoroughly

studied from a decision-theoretic perspective, both from the frequentist and Bayesian points of view, following the seminal work of Wald [1950].

In recent years, there has been an increasing interest in issues related with hypothesis testing problems in which hostile adversaries perturb the data observed by a decision-maker to confound her about the relevant hypothesis and gain some benefit. Examples come from the fields of adversarial signal processing, see Barni and Pérez-González [2013] for an introduction; adversarial classification, with the pioneer work in Dalvi et al. [2004]; adversarial medical testing, as undertaken by Singpurwalla et al. [2016]; and adversarial machine learning, see e.g. Tygar [2011]. These cover applications like online fraud detection, watermarking, survival analysis or spam detection, among many others.

Most attempts in these areas have focused on game-theoretic approaches to hypothesis testing, with the entailed common knowledge assumptions. For example, Barni and Tondi [2014] provide a framework focusing on zero-sum game-theoretic minimax procedures to hypothesis testing. This is not realistic since losses for various participants will be typically asymmetric, and, moreover, the beliefs and preferences of the adversary will not be readily available. Thus, key conditions of the commonly proposed techniques would not hold.

AHT shows the broad applicability of the ARA methodology. Bearing in mind that ARA appeared as a way forward to implement game theory to adversarial real-world cases, the resolution of new problems is vital for the development of ARA models so that they are understandable to the decision-maker and not too complex to be efficiently implemented. Thus, an ARA framework to AHT, along with some problem-specific solutions, is devised in this PhD thesis.

1.5 Research objectives

The aim in this PhD thesis is to provide new models in the context of ARA to solve certain problems identified in the current paradigm of (non-cooperative) game theory. Specifically, problems related to the common knowledge assumptions and the limited scope of solution concepts and rationalities associated with Nash equilibria and derived concepts. As two main lines of research are covered, the main objectives and secondary targets of interest will be indicated for both of them.

The first research line corresponds to the current research strategy in ARA, that is the development of a general ARA scheme to provide solutions to specific conflict situations. With this purpose, Section 1.4.1 presented different representation methods frequently used for non-cooperative games concluding that bi-agent influence diagrams result in useful structures for the ARA approach that need yet to be operationalised, justifying Objective O1.

O1. Develop representation methods in ARA. Representation methods used for (non-cooperative) games have not yet been fully assessed from the ARA perspective. Relatively simple ARA models have been developed, regarded as templates, but comprehensive adversarial problems still need to be studied. Thus, several subobjectives are:

(i) Provide computational schemes for general proper bi-agent influence diagrams from the ARA perspective.

(ii) Produce a computational environment to support the ARA methodology for bi-agent influence diagrams.

(iii) Address the use of the above methods in security related problems.

The second research line intends to provide tools to model different thinking strategies that may be associated with the adversaries. The discussion in Section 1.4.2, which introduced concept uncertainty and prevailing rationality paradigms bestowed on opponents, leads to Objective O2.

O2. Enhance the modelling of strategic reasoning. A distinctive element of ARA is the strategic reasoning modelling. Different rationality paradigms may be used for the adversaries and it is crucial to evaluate the soundness of the chosen strategies. In this case, the subobjectives are:

(i) Contemplate different solution concepts and how to handle them from the ARA perspective.

(ii) Procure methods to deal with concept uncertainty.

Finally, Section 1.4.3 reviewed earlier models that have been devised to deal with a variety of conflict situations encompassed within the scope of adversarial hypothesis testing, providing a set of specific applications of ARA and making for Objective O3. This objective is transversal in relation to the rest.

O3. Implement ARA models. All developed ARA models should be applied to real-world problems or numerical examples for testing purposes. Related subobjectives are:

(i) Take the efficiency of models into consideration.

(ii) Develop models which are understandable to decision-makers.

The objectives' development involves the use of a variety of mathematical tools related to the ARA paradigm, MAIDs, Bayesian inference and expert elicitation, among many others. In order to develop the different computational schemes, they will be implemented in R [2008] and MATLAB [2017].

1.6 Dissertation structure

An introduction and motivation of the proposed research has been provided in the present chapter, while the actual research is detailed in Chapters 2, 3 and 4. To wrap up, some conclusions and further work are discussed in Chapter 5. Supplementary materials consist of clarifying additional figures for Chapter 2 (Appendix A) and the relevant original `R` and `MATLAB` codes used for the numerical examples' resolution (Appendix B).

Specifically, Chapter 2 deals with Objective O1 describing how to support a decision-maker who faces an adversary with their joint problem modelled as a bi-agent influence diagram. Unlike previous solutions framed under a standard game-theoretic perspective, a decision-analytic methodology based on an ARA approach is provided using a schematic critical infrastructure protection problem for illustrative purposes.

With regard to Chapter 3, Objective O2 is undertaken. As such, concept uncertainty is studied under an ARA perspective, adding a new layer to the traditional risk analysis distinction between aleatory and epistemic uncertainties, when adversaries are present. Adversarial statistical decision theory is introduced with a foundational prospect and used to construct a leading example from the specific case of adversarial point estimation.

Persisting on the adversarial statistical decision theory defined in the previous chapter, the fundamental problem of AHT as representative of Objective O3 is considered in Chapter 4. In particular, a hypothesis testing situation in which an adversary aims at distorting the relevant data-process monitored by the decision-maker so as to confound her and achieve a certain benefit. An ARA solution to this problem is developed and its use extended to a batch acceptance context with partial observations of the data-process.

Finally, Chapter 5 summarises all conducted research and raises new questions derived from the work in Chapters 2, 3 and 4. In addition, (stochastic) differential games are suggested as a whole new set of problems that could be faced within the ARA paradigm.

Chapter 2

Bi-agent influence diagrams

This chapter provides an ARA algorithmic approach to solve general bi-agent adversarial problems using the BAIDs capacity to model complex interactions between both agents, taking advantage of the strategic relevance and s-reachability concepts, yet relaxing the common knowledge assumptions through the ARA methodology. Though many different rationality paradigms may be considered in the ARA framework [Ríos Insua, Banks and Ríos, 2016], the decision-maker is regarded as a level-2 thinker, in the Stahl and Wilson [1995] sense, so that she ponders how the adversary's strategy would adapt to her own one but presumes that he does not conduct likewise. The aim is to support her decision-making process, for which the opponent's intentions need to be forecast. For that, he is assumed to be an expected utility maximiser. The adversary's actions could be predicted by finding his maximum expected utility policy; however, the uncertainty in the decision-maker's assessments about the opponent's utilities and probabilities propagates to his optimal decisions providing instead probability distributions over the required forecasts.

In Section 2.1, IDs and BAIDs are defined and the problems that shall be dealt with presented, with a driving example in Critical Infrastructure Protection (CIP) being provided. Section 2.2 illustrates the key computational features of the proposal applied to that example. Section 2.3 first reviews ID reduction operations and strategic relevance concepts to then develop the general methodology, while Section 2.4 procures a numerical example. Section 2.5 conducts a recapitulation.

2.1 Formal definition

An ID is a directed graph representing a decision-making problem [Miller et al., 1976]. Nodes represent random variables (circles), decisions (squares) and utilities (hexagons). Arrows into chance or value nodes specify conditional dependence,

while dashed arrows into decision nodes indicate the available information when making that decision. For example, Figure 2.1 depicts the ID for a simple decision-making process in which a random event (S) occurs and an agent responds to it (D) perceiving some utility (u). In the ID, the dashed arrow emerging from chance node S to decision node D reflects that the agent makes her choice having observed the outcome of the random event. Besides, arrows coming out from S and D to value node u manifest that the decision-maker's utility depends on both the random outcome and her choice. The preceding terminology is based on Banks et al. [2015], whereas the remaining features are shared with standard IDs as characterised by Shachter [1986].

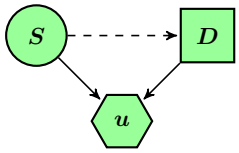


Figure 2.1: ID example

In opposition to decision trees as tools to perform decision analysis, IDs are compact, emphasising the relationships among variables, and yet represent complete probabilistic descriptions of the entailed decision-making problems. This results in substantial reductions in memory requirements. However, though being equivalent representation methods, IDs correspond to symmetric decision trees, so they cannot benefit from savings achieved through asymmetric processing. A comprehensive introduction to IDs may be found in Howard and Matheson [2005].

A BAID essentially consists of coupled IDs, one for the decision-maker and one for her adversary, possibly with shared chance nodes and some links between both agents' decision nodes. Figure 1.1 in Section 1.4.1 presented the BAIDs for five template models. In these, one can observe several chance, decision and utility nodes, corresponding to the decision-maker's (green) and her adversary's (red) problems, respectively. Bi-colour nodes represent common chance nodes, in the sense that such uncertainties are relevant in both agents' decision-making. However, they may entertain different probability models over such nodes which, as mentioned, will not be common knowledge.

Schemes for implementing ARA within such templates may be seen in Ríos and Ríos Insua [2012] and Ríos Insua et al. [2013]. Still, such stylised settings may not be sufficient to cope with the complexities of many real problems. As an illustration, consider the BAID in Figure 2.2, which shall be used to outline and demonstrate the methodology. It refers to a scenario related to CIP. In it, the incumbent authorities (decision-maker, defender) have to decide whether or not to increase the infrastructure's protection against terrorist attacks. To this aim, they could deploy additional measures (D_1) by e.g. reinforcing security controls on people and

items. Meanwhile, terrorists (opponent, attacker) might be pondering over infiltrating within the infrastructure (A_1) to gain intelligence for future attacks. Decisions D_1 and A_1 are simultaneous and unknown to the respective adversary while being made. The attacker then observes D_1 and springs to action, fearing that new countermeasures might be added soon. He chooses his attack (A_2) consisting of a direct attempt to cause damage to the infrastructure. The interaction between decisions D_1, A_1 and A_2 would yield a random outcome (S_1) describing the consequences of the attack. Depending on such consequences and the attacker's action A_2, the defender will implement recovery measures (D_2). Finally, the combined effect of decisions A_2 and D_2 along with outcome S_1 would determine a random effect (S_2), which determines the impact of the recovery effort. For each agent, there is a value node which assesses all their consequences, respectively represented by u_D and u_A. In particular, the defender's utility u_D (respectively, the attacker's utility u_A) will depend on her decisions D_1 and D_2 (respectively, his decisions A_1 and A_2) and the results S_1 and S_2. This, or similar sequences of defence-attack movements, could be repeated across time, spanning over several planning periods.

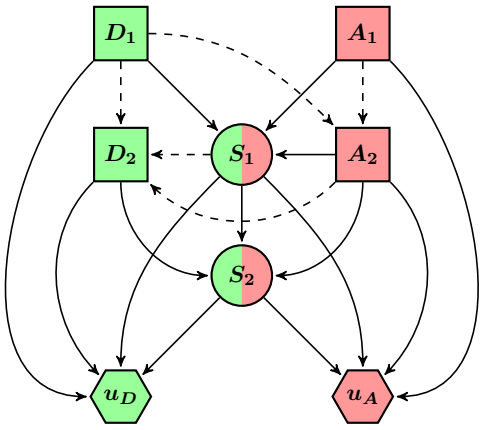


Figure 2.2: ARA modelling of the CIP bi-agent problem

Note that, within the general layout of the BAID in Figure 2.2, patterns of defence and attack movements from those earlier described in Section 1.4.1 as basic templates may be identified. In particular, nodes D_1–A_1 correspond to the Sim. D-A template, in which both agents decide their move without knowing the action chosen by each other. Similarly, nodes D_1–A_2 reproduce the backbone structure of a Seq. D-A template, in which the defender first chooses her action and then, having observed it, the attacker decides his own move. Finally, nodes D_1–A_2–D_2 may be regarded as drawn from a Seq. D-A-D template. Detecting such patterns within the global layout of the BAID will be key in the proposed solution strategy.

Though general adversarial problems for two agents interacting with each other over time are to be considered, not all BAIDs conceivable as the mere junction of two IDs (one for the decision-maker, one for her adversary) with several shared

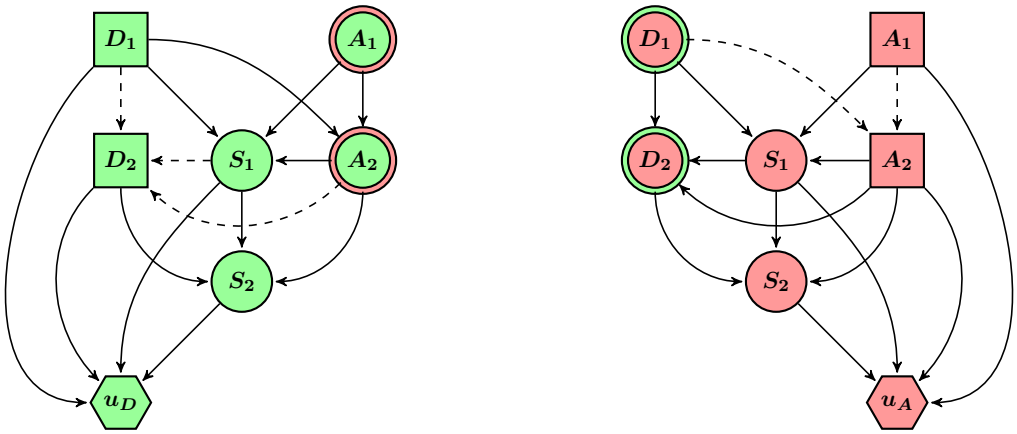
chance nodes and arrows linking their decision nodes will be dealt with. To ensure consistency between the informational structure and the ordering of the agents' analysis, terminology from Shachter [1986] is drawn, extended to the bi-agent case.

IDs are said to be *proper* if they are an unambiguous representation of a single player's perspective of a decision-making problem. Within them, a relevant subset is that of *regular* IDs which satisfy that: (i) the directed graph has no cycles; (ii) the value node, if present, is unique and has no successors; and (iii) there is a directed path that contains all of the decision nodes. In the ARA setting, a proper BAID will be an acyclic directed graph over chance, decision and value nodes, where chance nodes can be shared by both agents, such that, from each player's perspective, it is a regular ID. Essentially, if two decisions are simultaneous, there must be no directed path between them.

To check whether a BAID satisfies this constraint, an ID is obtained for each agent by deleting their opponent's value node and converting his/her decision nodes into chance ones. Chance nodes that relate only to the other player are also eliminated. Decision nodes owned by the other agent may become barren ones (nodes with no successors) and, thus, be removed. Each player's ID, $\mathcal{D}$ for the decision-maker and $\mathcal{A}$ for her adversary, must then define a total order of decisions and a corresponding partial order of chance nodes. The total order for the decision-maker is associated with the chronological order of her m decisions: $D_1 \rightarrow^{\mathcal{D}} D_2 \rightarrow^{\mathcal{D}} \cdots \rightarrow^{\mathcal{D}} D_m$. This induces a partition of the set $C^{\mathcal{D}}$ of chance nodes relevant for her: (i) $C_0^{\mathcal{D}}$, consists of those random events that she is acquainted with when making her first decision D_1; (ii) $C_i^{\mathcal{D}}$, contains those chance nodes whose values she observes between her decisions D_i and D_{i+1}, for $i = 1, \ldots, m-1$; and, finally, (iii) $C_m^{\mathcal{D}}$, are those chance nodes that she does not perceive before making all of her decisions. Some of these $C_i^{\mathcal{D}}$ sets might be empty. This partition defines a partial order over decision and chance nodes $C_0^{\mathcal{D}} \prec^{\mathcal{D}} D_1 \prec^{\mathcal{D}} \cdots \prec^{\mathcal{D}} C_{m-1}^{\mathcal{D}} \prec^{\mathcal{D}} D_m \prec^{\mathcal{D}} C_m^{\mathcal{D}}$, inducing an information structure within the temporal order of decisions that specifies the information known by the decision-maker at the time she makes each decision. Similarly, a partial order $C_0^{\mathcal{A}} \prec^{\mathcal{A}} A_1 \prec^{\mathcal{A}} \cdots \prec^{\mathcal{A}} C_{n-1}^{\mathcal{A}} \prec^{\mathcal{A}} A_n \prec^{\mathcal{A}} C_n^{\mathcal{A}}$ may be defined for the n decisions $A_1 \rightarrow^{\mathcal{A}} A_2 \rightarrow^{\mathcal{A}} \cdots \rightarrow^{\mathcal{A}} A_n$ to be made by her adversary and the incumbent set $C^{\mathcal{A}}$ of his chance nodes. $D^{\mathcal{D}}$ (respectively, $D^{\mathcal{A}}$) shall designate the set of decision nodes in $\mathcal{D}$ (respectively, $\mathcal{A}$). Note that, except for barren nodes which are assumed to have been eliminated, $D^{\mathcal{D}} \subset C^{\mathcal{A}}$ and $D^{\mathcal{A}} \subset C^{\mathcal{D}}$, as adversarial actions are uncertain to the opposing agent. It could be the case that $C^{\mathcal{D}} \cap C^{\mathcal{A}} \neq \emptyset$ due to some chance nodes being shared.

The two associated IDs represent, respectively, the decision-maker's and her adversary's viewpoint of their problems. As an example, consider the CIP model in Figure 2.2. When the defender's problem is addressed, the attacker's decisions at nodes A_1 and A_2 are treated as uncertain from the defender's perspective. By substituting $\boxed{A}$ with $\textcircled{\textcircled{A}}$ this is reflected in the ID in Figure 2.3a, where the attacker's decision nodes have been converted into defender's chance nodes. Note, however, that the

defender will observe the outcome of (for her) chance node A_2 prior to making her own decision D_2. When the defender tries to solve her problem, she will need to analyse the problem from the attacker's stance, Figure 2.3b. Revealed by replacing D with D, decision nodes D_1 and D_2 are now chance nodes for the attacker, as he is uncertain about the defender's intentions. Similarly, the attacker will perceive the result of (for him) chance node D_1 prior to deciding upon A_2.



(a) Defender's problem

(b) Attacker's problem

Figure 2.3: Both agents' IDs in the CIP model

2.2 Algorithmic approach

A description on how to deal with the CIP example in Figure 2.2 is provided as a motivation for the general computational strategy proposed in Section 2.3.1. First, the defender's problem $\mathcal{D}$, depicted in Figure 2.3a, is solved. Her utility function is denoted by $u_D(d_1, d_2, s_1, s_2)$; her beliefs over the attack S_1 and recovery S_2 outcomes, respectively, by $p_D(s_1 \,|\, d_1, a_1, a_2)$ and $p_D(s_2 \,|\, d_2, a_2, s_1)$; and her probability distributions over the attacker's decisions A_1 and A_2, respectively, by $p_D(a_1)$ and $p_D(a_2 \,|\, d_1, a_1)$. The ID is evaluated backwards following the approach in Shachter [1986]:

$\mathcal{D}$1. Remove chance node S_2, computing the expected utilities

$$\psi_D(d_1, d_2, a_2, s_1) = \int u_D(d_1, d_2, s_1, s_2)\, p_D(s_2 \,|\, d_2, a_2, s_1)\, \mathrm{d}s_2.$$

$\mathcal{D}$2. Eliminate decision node D_2, determining and storing the optimal action, given d_1, a_2 and s_1,

$$d_2^*(d_1, a_2, s_1) = \arg\max_{d_2}\ \psi_D(d_1, d_2, a_2, s_1),$$

and recording the optimal expected utility

$$\psi_D(d_1, a_2, s_1) = \max_{d_2} \ \psi_D(d_1, d_2, a_2, s_1).$$

$\mathcal{D}$3. Take out chance node S_1, calculating the expected utilities

$$\psi_D(d_1, a_1, a_2) = \int \psi_D(d_1, a_2, s_1)\, p_D(s_1 \,|\, d_1, a_1, a_2)\, \mathrm{d}s_1.$$

$\mathcal{D}$4. Withdraw chance (for the defender) node A_2, estimating the expected utilities

$$\psi_D(d_1, a_1) = \int \psi_D(d_1, a_1, a_2)\, p_D(a_2 \,|\, d_1, a_1)\, \mathrm{d}a_2.$$

$\mathcal{D}$5. Delete chance (for the defender) node A_1, obtaining the expected utilities

$$\psi_D(d_1) = \int \psi_D(d_1, a_1)\, p_D(a_1)\, \mathrm{d}a_1.$$

$\mathcal{D}$6. Finally, reduce node D_1, finding the maximum expected utility decision

$$d_1^* = \arg\max_{d_1} \ \psi_D(d_1).$$

The defender's optimal strategy would be to choose d_1^* at node D_1 and, later, $d_2^*(d_1, a_2, s_1)$ at node D_2.

While the assessments of $u_D(d_1, d_2, s_1, s_2)$, $p_D(s_1 \,|\, d_1, a_1, a_2)$ and $p_D(s_2 \,|\, d_2, a_2, s_1)$ are standard in the decision analysis practice [French and Ríos Insua, 2000], those of $p_D(a_1)$ and $p_D(a_2 \,|\, d_1, a_1)$ require strategic thinking. To anticipate her adversary's movements, the defender may analyse the attacker's problem $\mathcal{A}$, shown in Figure 2.3b. As acknowledged, the defender is considered to be a level-2 thinker: her uncertainty about the attacker's decisions, which stems from her uncertainty about the attacker's utilities and probabilities, is modelled assuming he is an expected utility maximiser. Therefore, the defender needs to assess his utility function $u_A(a_1, a_2, s_1, s_2)$; his beliefs over the attack S_1 and recovery S_2 outcomes, respectively, $p_A(s_1 \,|\, d_1, a_1, a_2)$ and $p_A(s_2 \,|\, d_2, a_2, s_1)$; as well as his distributions over her own decisions D_1 and D_2, respectively, $p_A(d_1)$ and $p_A(d_2 \,|\, d_1, a_2, s_1)$. However, in general, she will not know the attacker's utilities and probabilities since common knowledge is not available. Assume she may model such uncertainty about them through random utilities and probabilities, which may be described as $F \sim \big(U_A(a_1, a_2, s_1, s_2), P_A(s_1 \,|\, d_1, a_1, a_2), P_A(s_2 \,|\, d_2, a_2, s_1), P_A(d_1), P_A(d_2 \,|\, d_1, a_2, s_1)\big)$. To obtain the (random) optimal alternatives for the attacker, the uncertainty in F would then be propagated:

$\mathcal{A}$1. Remove chance node S_2, computing the attacker's (random) expected utilities

$$\Psi_A(d_2, a_1, a_2, s_1) = \int U_A(a_1, a_2, s_1, s_2)\, P_A(s_2 \,|\, d_2, a_2, s_1)\, \mathrm{d}s_2.$$

$\mathcal{A}$2. Eliminate chance (for the attacker) node D_2, estimating the attacker's (random) expected utilities

$$\Psi_A(d_1, a_1, a_2, s_1) = \int \Psi_A(d_2, a_1, a_2, s_1)\, P_A(d_2 \,|\, d_1, a_2, s_1)\, \mathrm{d}d_2.$$

$\mathcal{A}$3. Take out chance node S_1, calculating the attacker's (random) expected utilities

$$\Psi_A(d_1, a_1, a_2) = \int \Psi_A(d_1, a_1, a_2, s_1)\, P_A(s_1 \,|\, d_1, a_1, a_2)\, \mathrm{d}s_1.$$

$\mathcal{A}$4. Withdraw decision node A_2, determining the attacker's (random) optimal decisions, given d_1 and a_1,

$$A_2^*(d_1, a_1) = \arg\max_{a_2}\ \Psi_A(d_1, a_1, a_2),$$

and recording the (random) optimal expected utilities

$$\Psi_A(d_1, a_1) = \max_{a_2}\ \Psi_A(d_1, a_1, a_2).$$

$\mathcal{A}$5. Delete chance (for the attacker) node D_1, obtaining the attacker's (random) expected utilities

$$\Psi_A(a_1) = \int \Psi_A(d_1, a_1)\, P_A(d_1)\, \mathrm{d}d_1.$$

$\mathcal{A}$6. Reduce decision node A_1, finding the attacker's (random) optimal decision

$$A_1^* = \arg\max_{a_1}\ \Psi_A(a_1).$$

The optimal (random) attacks would be A_1^* and $A_2^*(d_1, a_1)$.

Finally, the defender's probability $p_D(a_2 \,|\, d_1, a_1)$ over attack A_2, conditional on her first defence decision d_1 and the attacker's first move a_1, which she needs in step $\mathcal{D}4$, would be given by

$$\int_{-\infty}^{a_2} p_D(a_2 = y \,|\, d_1, a_1)\, \mathrm{d}y = \Pr(A_2^*(d_1, a_1) \leq a_2).$$

Similarly, her probability $p_D(a_1)$ over attack A_1, which she requires in step $\mathcal{D}5$, is given by

$$\int_{-\infty}^{a_1} p_D(a_1 = x)\, \mathrm{d}x = \Pr(A_1^* \leq a_1).$$

This information would be incorporated to the defender's problem to obtain her optimal defence. Both distributions can be approximated through Monte Carlo simulation.

Given the sequentiality of the problem, with alternation of simultaneous and sequential decisions by both agents, it may be solved stepwise, scheduling optimisation stages from the defender and simulation stages from the attacker, to get the requested distributions in the defender's problem). One could think of tackling first all the simulation stages and then the optimisation ones, but switching between both problems allows to better apportion the uncertainty around the defender's optimal decisions, as suggested in Merrick and Parnell [2011]. In this manner, the defender's distributions over the attacker's probabilities for her decisions can be better assessed, in line with Fermitisation strategies in structured expert judgement methodologies [Tetlock and Gardner, 2015]. Indeed, schemes $\mathcal{D}$ ($\mathcal{D}1$–$\mathcal{D}6$) and $\mathcal{A}$ ($\mathcal{A}1$–$\mathcal{A}6$) may be combined in different ways into a single one which jumps from $\mathcal{D}$ to $\mathcal{A}$ steps and backwards, e.g. as in the following scheme $\mathcal{E}$:

$\mathcal{E}1$. Perform $\mathcal{D}1$. The ingredients $u_D(d_1, d_2, s_1, s_2)$ and $p_D(s_2 \,|\, d_2, a_2, s_1)$ are available from the defender. Then, compute the expected utilities in $\mathcal{D}1$.

$\mathcal{E}2$. Complete $\mathcal{D}2$ to remove D_2 and find optimal decision $d_2^*(d_1, a_2, s_1)$, given d_1, a_2 and s_1.

$\mathcal{E}3$. Implement $\mathcal{D}3$, where $p_D(s_1 \,|\, d_1, a_1, a_2)$ is also provided by the defender. Thus, calculate the expected utilities in $\mathcal{D}3$.

$\mathcal{E}4$. At step $\mathcal{D}4$ $p_D(a_2 \,|\, d_1, a_1)$ is needed. Since the defender lacks it, switch to problem $\mathcal{A}$. Recurring to $U_A(a_1, a_2, s_1, s_2)$, $P_A(s_1 \,|\, d_1, a_1, a_2)$, $P_A(s_2 \,|\, d_2, a_2, s_1)$ and $P_A(d_2 \,|\, d_1, a_2, s_1)$, execute steps $\mathcal{A}1$ to $\mathcal{A}3$ and then, $\mathcal{A}4$, getting $p_D(a_2 \,|\, d_1, a_1)$ from $\Pr(A_2^*(d_1, a_1) \leq a_2)$. Then finalise $\mathcal{D}4$.

$\mathcal{E}5$. Step $\mathcal{D}5$ requires $p_D(a_1)$. The defender does not have it, so turn again to problem $\mathcal{A}$. Using $P_A(d_1)$, realise $\mathcal{A}5$. Then, conduct $\mathcal{A}6$, obtaining $p_D(a_1)$ from $\Pr(A_1^* \leq a_1)$, and conclude $\mathcal{D}5$.

$\mathcal{E}6$. Finally, carry out step $\mathcal{D}6$ to eliminate D_1 and find optimal decision d_1^*.

Essentially, this scheme solves as many $\mathcal{D}$ steps (with standard ID reduction operations) as possible until some assessment from the attacker is needed to solve another $\mathcal{D}$ step. Then, as few steps from problem $\mathcal{A}$ are solved, with ID reduction operations modified to take into account the uncertainty about the attacker's utilities and probabilities, until the required attacker's assessment is obtained. At this point one jumps back to the defender's problem and proceeds as above until all defender's decision nodes have been reduced.

Deciding when to jump from problem $\mathcal{D}$ to problem $\mathcal{A}$, and backwards, is relatively simple to perform by hand, but can be messy from an algorithmic point of view. This may be achieved through the use of the relevance and component graphs of the BAID, described in Koller and Milch [2003] and reviewed in Section 2.3.2. The relevance graph for the CIP example is shown in Figure 2.4, together with the so-called maximal Strongly Connected Components (SCCs, delimited by dashed lines) $\{D_1, A_1\}$, $\{A_2\}$ and $\{D_2\}$ which define the component graph. This graph induces a topological ordering among the defender's and attacker's decisions which, in our example, is unique: $\{D_2\} \rightarrow \{A_2\} \rightarrow \{D_1, A_1\}$. Note that this encompasses the identified simultaneous D_1–A_1 block, as well as the sequential D_1–A_2 and A_2–D_2 blocks, complemented with the decision sequences $D_1 \rightarrow^{\mathcal{D}} D_2$ (for the defender) and $A_1 \rightarrow^{\mathcal{A}} A_2$ (for the attacker).

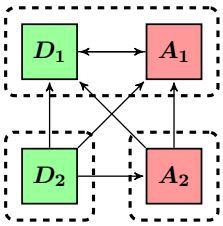


Figure 2.4: Relevance graph for the CIP example

As stated, scheme $\mathcal{E}$ is difficult to generalise from an algorithmic point of view. However, using the topological ordering induced by the component graph, as well as the decision sequences for each agent, a more general and systematic approach, designated $\mathcal{G}$, may be designed:

$\mathcal{G}$1. Tackle the first SCC: $\{D_2\}$. It consists only of one of the defender's decision nodes. Using her utility $u_D(d_1, d_2, s_1, s_2)$ and probability $p_D(s_2 \,|\, d_2, a_2, s_1)$, complete steps $\mathcal{D}1$ and $\mathcal{D}2$ to eliminate decision D_2.

$\mathcal{G}$2. Deal with the second SCC: $\{A_2\}$. It comprises an attacker's decision node. Through assessments $U_A(a_1, a_2, s_1, s_2)$, $P_A(s_1 \,|\, d_1, a_1, a_2)$, $P_A(s_2 \,|\, d_2, a_2, s_1)$ and $P_A(d_2 \,|\, d_1, a_2, s_1)$, get $p_D(a_2 \,|\, d_1, a_1)$ from $\Pr(A_2^*(d_1, a_1) \leq a_2)$ by performing steps $\mathcal{A}1$ to $\mathcal{A}4$.

$\mathcal{G}$3. Take up the remaining SCC: $\{D_1, A_1\}$. This combines defender's and attacker's decision nodes. Begin by solving the attacker's decision node A_1. Applying $P_A(d_1)$, carry out $\mathcal{A}5$. Finally, achieve $\mathcal{A}6$, obtaining $p_D(a_1)$ from $\Pr(A_1^* \leq a_1)$.

$\mathcal{G}$4. Finish with last SCC: $\{D_1, A_1\}$. Only decision node D_1 belonging to the defender awaits to be reduced. Probability $p_D(s_1 \,|\, d_1, a_1, a_2)$ is available to the defender, so conduct $\mathcal{D}3$. At step $\mathcal{D}4$ $p_D(a_2 \,|\, d_1, a_1)$ is need, but it was already

found at $\mathcal{G}2$, so it may be accomplished. In the same way, step $\mathcal{D}5$ requires $p_D(a_1)$ which was estimated at $\mathcal{G}3$. To conclude, execute $\mathcal{D}6$ to eliminate D_1.

Appendix A graphically represents the implementation of solution scheme $\mathcal{G}$, which is now generalised.

2.3 General computational strategy

A general methodology for solving BAIDs through ARA is described in this section. Two agents, the decision-maker and her adversary, are assumed to deal with a problem which may be modelled with a proper BAID, as indicated in Section 2.1. First, the basic BAID reduction operations are presented. Then, a brief review of the necessary strategic relevance concepts is provided. Finally, an algorithm for ARA support to an agent in a problem modelled as a BAID is introduced.

2.3.1 BAID reduction operations

As in Section 2.2, reduction operations referring to $\mathcal{D}$ steps and $\mathcal{A}$ steps are distinguished. Those in relation to problem $\mathcal{D}$ (*D-reductions*) correspond to standard ID reduction operations, as in Shachter [1986], which shall be called *D-barren node removal*, *D-arc inversion*, *D-chance node removal* and *D-decision node removal*. A *D-barren node* is a chance or decision node without successors in problem $\mathcal{D}$.

The operations in relation to problem $\mathcal{A}$ (*A-reductions*) must take into account the uncertainty about the adversary's utilities and probabilities. Therefore, they need to be modified, except for the *A-barren node removal*, which coincides. With no loss of generality, it may be presumed that all involved random utilities U_A and probabilities P_A are defined over a common probability space $(\Omega, \mathcal{F}, \mathcal{P})$ [Chung, 1968], with atomic elements ω (single events in the sample space). Thus, when invoking them in reference to ω, they shall be designated as U_A^ω and P_A^ω, respectively. The remaining notation is adapted from Shachter [1986] where: superscripts *old* and *new* refer to an element, respectively, before or after the BAID transformation; and, for any node i, $C(i)$ designates its conditional predecessors (chance and value nodes), $I(i)$ its informational predecessors (decision nodes) and $S(i)$ its direct successors. Initially, $\Psi_A^\omega(x_{C(v)}) = U_A^\omega(x_{C(v)})$ where v refers to the value node.

- *A-Arc inversion.* An arc (i, j) between chance (for the attacker) nodes i and j, such that there is no other directed path between i an j, may be replaced by the arc (j, i). Node inheritance is as in conventional IDs. (Random) probability

assessments are changed by applying Bayes' formula parametrically so that

$$P_A^{\omega\,new}(x_j|x_{C^{new}(j)}) = \int P_A^{\omega\,old}(x_j|x_{C^{old}(j)})\, P_A^{\omega\,old}(x_i|x_{C^{old}(i)})\,\mathrm{d}x_i,$$

$$P_A^{\omega\,new}(x_i|x_{C^{new}(i)}) = \frac{P_A^{\omega\,old}(x_j|x_{C^{old}(j)})\, P_A^{\omega\,old}(x_i|x_{C^{old}(i)})}{P_A^{\omega\,new}(x_j|x_{C^{new}(j)})}.$$

- *A-Chance node removal.* A chance (for the attacker) node i whose only successor is the value node v may be removed, with v inheriting the conditional predecessors of i. Note, however, that expectations have to be taken parametrically so as to obtain (random) expected utilities. Prior to node i reduction, there is a (random) expected utility $\Psi_A^{\omega\,old}(x_{C^{old}(v)})$ related to each combination $x_{C^{old}(v)}$ of values for predecessors of v. After the reduction, a new (random) expected utility is associated with v defined by

$$\Psi_A^{\omega\,new}(x_{C^{new}(v)}) = \int \Psi_A^{\omega\,old}(x_{C^{old}(v)})\, P_A^{\omega}(x_i|x_{C(i)})\,\mathrm{d}x_i.$$

- *A-Decision node removal.* Assuming no barren nodes, a decision (for the attacker) node i which is a predecessor of the value node v and whose predecessors include those of v may be reduced, by computing the expected utility of the (random) optimal alternatives, conditional on the values of its predecessors. Node inheritance is as in conventional IDs. (Random) optimal alternatives are stored through

$$A_i^{\omega *}(x_{C^{new}(v)}) = \arg\max_{x_i}\ \Psi_A^{\omega\,old}(x_i, x_{C^{new}(v)}),$$

  whereas their (random) expected utilities are

$$\Psi_A^{\omega\,new}(x_{C^{new}(v)}) = \max_{x_i}\ \Psi_A^{\omega\,old}(x_i, x_{C^{new}(v)}).$$

To check that the above A-reductions are correctly defined, one just needs to realise that, when conditioning on the atomic element ω of the underlying σ-algebra, problem $\mathcal{A}$ behaves as an ordinary ID parametrised by ω. Therefore, standard ID reduction operations (as proved by Shachter [1986]) may be applied to each of the parametrised IDs, conforming the characterised A-reductions.

The procedure that will be used to decide which reduction operation to apply at each step, whether in problem $\mathcal{D}$ or $\mathcal{A}$, is that provided by Shachter [1986], which runs in quadratic time in the number of nodes in the BAID. For the sake of completeness, it is next included.

Algorithm 1 BAIDs: Shachter's ID reduction procedure

Data: Proper ID $\mathcal{I}$ with $C(v) \neq \emptyset$.

If $\exists i \in C \cap C(v)$ such that $S(i) = \{v\}$ **then**
  Remove chance node i.
Else If $\exists i \in D \cap C(v)$ such that $C(v) \subset I(i) \cup \{i\}$ **then**
  Remove decision node i and all resulting barren nodes.
Else
  Find $i \in C \cap C(v)$ such that $D \cap S(i) = \emptyset$.
  While $C \cap S(i) \neq \emptyset$ **do**
    Find $j \in C \cap S(i)$ such that there is no other directed (i, j) path.
    Reverse arc (i, j).
  End While
  Remove chance node i.
End If

2.3.2 Strategic relevance concepts

The concept of strategic relevance will help in identifying when to jump from problem $\mathcal{D}$ to problem $\mathcal{A}$, and backwards, as well as the structural blocks within the BAID. The basic concepts are briefly recalled, with further details in Koller and Milch [2003].

A decision node N_j is *strategically relevant* for decision node N_i if and only if the decision made at node N_j is need to be known to make the decision at node N_i. It is also said that N_i (strategically) relies on N_j. A graphical criterion for detecting strategic relevance is provided by *s-reachability*: a decision node N_j is s-reachable from a decision node N_i in a BAID if and only if the utility node u associated with N_i verifies that if a new predecessor N_p were added to N_j, there would be an active path from N_p to u given $N_i \cup C(N_i) \cup I(N_i)$ viewing the BAID as a Bayesian network. As proved by Koller and Milch [2003], N_j is s-reachable from N_i if N_i relies on N_j, though N_i does not rely on N_j in every BAID if N_j is s-reachable from N_i, as probabilities and utilities in the BAID may be chosen in such a way that the influence of one decision rule on another does not manifest itself. However, the latter is true for some BAID with the same graph structure, so it is safe to assume that strategic relevance and s-reachability are equivalent in the general case.

Based on the s-reachability concept, the *relevance graph* for the BAID is built, which is a directed graph whose nodes are the decision nodes (of both agents) in

the BAID and contains an arc $N_j \to N_i$ if and only if N_j is s-reachable for N_i. Note that, according to this definition, all strategic reliance relations between decision nodes are being manifested in the relevance graph and that every arc provides a potential strategic reliance relation in the BAID. If two decision variables N_i and N_j within the relevance graph are s-reachable from each other, the graph is declared *cyclic*. Otherwise, it is called *acyclic*. The sets $\mathcal{N}$ of nodes such that for every pair of them $N_i, N_j \in \mathcal{N}$ there exists a directed path from N_i to N_j, are named Strongly Connected Components (SCCs). SCCs which are not a strict subset of another one are designated maximal. When the relevance graph is acyclic, the SCCs are singletons. The *component graph*, which is acyclic, has the maximal SCCs as nodes and includes an arc between two of them if and only if one of the nodes in the predecessor component is s-reachable from one of the nodes in the successor component.

For any BAID the relevance graph can be constructed, e.g. using an algorithm such as Bayes-Ball [Shachter, 1998], which allows to determine the set of nodes which are s-reachable for each decision node and was used to build Figure 2.4. This algorithm runs in linear time in the number of nodes in the BAID. By applying it to each decision node, the relevance graph is obtained in quadratic time in the number of nodes in the BAID.

If all decisions made by the decision-maker and her adversary have to be performed sequentially, the relevance graph is acyclic and a complete topological ordering of the decision nodes may be found, $N_1 \to N_2 \to \cdots \to N_{m+n}$, although not necessarily unique, such that if N_j is s-reachable from N_i, then N_i precedes N_j. This ordering corresponds to the reverse sequence of decisions that both agents make during their interaction, corresponding to sequential Defend-Attack and Attack-Defend blocks, possibly interlinked by sequences of Attack or Defence decisions. As an example, one could have a sequential Defend-Attack-Attack-Defend-Attack-Defend model.

However, it could happen that certain pairs of decisions are made simultaneously, one by each agent, without prior knowledge of the other's action, corresponding to simultaneous Defend-Attack blocks. Such paired decisions are s-reachable from each other, preventing a precedence ordering between them and, consequently, making the relevance graph cyclic. Should that be the case, the component graph from the SCCs would be built to provide the required reduction mechanism. This task may be accomplished in linear time in the number of nodes in the BAID.

2.3.3 Computational strategy: The acyclic case

Algorithm 2 describes a general computational strategy to evaluate proper BAIDs, assuming that the relevance graph is acyclic. In this case, any (if multiple) of the topological orderings of all decision nodes in the BAID is found. Then, the decision

nodes at each ID are eliminated, according to the sequence provided by it, using the necessary D-reductions (for the defender's ID, $\mathcal{D}$) or A-reductions (for the attacker's ID, $\mathcal{A}$).

As mentioned, the defender is assumed to be a level-2 thinker. The aim is to determine which the defender's decisions should be. A more complete response, however, would provide information about those of the attacker through their likelihood. Thus, the algorithm will solve all decision nodes, regardless of them belonging to the defender or the attacker.

Algorithm 2 BAIDs: General computational strategy – Acyclic case

Data: BAID $\mathcal{B}$ with acyclic relevance graph; a topological ordering $N_1, \ldots, N_{m+n}$ of the relevance graph for $\mathcal{B}$; the associated IDs $\mathcal{D}$ for the defender and $\mathcal{A}$ for the attacker.

1: **For** $i = 1$ **to** $m + n$ **do**
2: **If** $N_i \in D^{\mathcal{D}}$ **then**
3: **While** $N_i \in \mathcal{D}$ **do**
4: Apply Algorithm 1 to $\mathcal{D}$ using D-reductions.
5: **End While**
6: **Else**
7: **While** $N_i \in \mathcal{A}$ **do**
8: Apply Algorithm 1 to $\mathcal{A}$ using A-reductions.
9: **End While**
10: **End If**
11: **End For**

Algorithm 2 removes the decision nodes one by one, following the topological ordering obtained from the relevance graph, until no node remains to be eliminated. Shachter's procedure (Algorithm 1) guarantees the application of successive reductions to the corresponding ID until a decision node is withdrawn and the next decision node may be targeted. Due to the proposed ordering, while reducing any of the problems $\mathcal{D}$ or $\mathcal{A}$ to remove the incumbent decision node, none of the remaining decision nodes is taken out and, thus, Algorithm 2 is well defined. As Shachter's procedure runs in quadratic time in the number of nodes in the BAID, so does Algorithm 2, which essentially performs such procedure twice, once for ID $\mathcal{D}$ and once for ID $\mathcal{A}$, though intertwined. This discussion may be formalised as follows:

Proposition 2.1. *For a problem modelled as a proper BAID with acyclic relevance graph, Algorithm 2 concludes in quadratic time in the number of nodes in the BAID providing the defender's optimal alternatives in her decision nodes based on the forecasts of the attacker's random optimal choices in his decision nodes.*

2.3.4 Computational strategy: The cyclic case

The above approach, however, will not always work. Problems will arise when some SCCs are not singletons, as in Figure 2.4 with $\{D_1, A_1\}$. There is a need to generalise the arguments in Section 2.2 for scheme $\mathcal{G}$, as done in Algorithm 3.

Algorithm 3 BAIDs: General computational strategy – Cyclic case

Data: BAID $\mathcal{B}$; a topological ordering $\mathcal{N}_1, \ldots, \mathcal{N}_r$ of the component graph derived from the relevance graph for $\mathcal{B}$; the associated IDs $\mathcal{D}$ for the defender and $\mathcal{A}$ for the attacker; the decision sequences $D_1, \ldots, D_m$ and $A_1, \ldots, A_n$, relative to $\mathcal{D}$ and $\mathcal{A}$.

For $i = 1$ **to** r **do**
  While $\mathcal{N}_i \cap D^{\mathcal{A}} \neq \emptyset$ **do**
    Find $j = \max\{k \mid A_k \in \mathcal{N}_i\}$.
    While $A_j \in \mathcal{A}$ **do**
      Apply Algorithm 1 to $\mathcal{A}$ using A-reductions.
    End While
  End While
  While $\mathcal{N}_i \neq \emptyset$ **do**
    Find $j = \max\{k \mid D_k \in \mathcal{N}_i\}$.
    While $D_j \in \mathcal{D}$ **do**
      Apply Algorithm 1 to $\mathcal{D}$ using D-reductions.
    End While
  End While
End For

Algorithm 3 solves the SCCs one by one, following the topological ordering obtained from the component graph, until none of them remains to be reduced. For each SCC, the attacker's decision nodes are tackled first in their reverse decision sequence so that the defender can best assess her predictions over the attacker's decisions. Then, the remaining decisions in the SCC are removed, which will correspond to the defender. As in Algorithm 2, ID reductions are performed until the targeted decision node removal is achieved while no other decision node is withdrawn, ensuring that the algorithm is well defined. As in Section 2.3.3, Algorithm 3 runs in quadratic time in the number of nodes in the BAID. An equivalent result to Proposition 2.1 may be formulated:

Proposition 2.2. *For a problem modelled as a proper BAID with cyclic relevance graph, Algorithm 3 concludes in quadratic time in the number of nodes in the BAID*

providing the defender's optimal alternatives in her decision nodes based on the forecasts of the attacker's random optimal choices in his decision nodes.

2.3.5 Computational strategy: The general approach

Both algorithms are now combined into a single one. The defender's and attacker's IDs are first built. If both are proper, the relevance graph is obtained. If this is acyclic, Algorithm 2 is followed. Otherwise, Algorithm 3 is adopted.

Algorithm 4 BAIDs: General computational strategy

Data: BAID $\mathcal{B}$.

1: Build ID $\mathcal{D}$ associated with $\mathcal{B}$ for the defender.
2: **If** $\mathcal{D}$ is not a proper ID **then**
3: **Break.**
4: **Else**
5: Eliminate all barren nodes from $\mathcal{D}$.
6: **End If**
7: Build ID $\mathcal{A}$ associated with $\mathcal{B}$ for the attacker.
8: **If** $\mathcal{A}$ is not a proper ID **then**
9: **Break.**
10: **Else**
11: Eliminate all barren nodes from $\mathcal{A}$.
12: **End If**
13: Build $\mathcal{R}$ the relevance graph for $\mathcal{B}$.
14: **If** $\mathcal{R}$ is acyclic **then**
15: Find a topological ordering $N_1, \ldots, N_{m+n}$ of $\mathcal{R}$.
16: Apply Algorithm 2.
17: **Else**
18: Find a topological ordering $\mathcal{N}_1, \ldots, \mathcal{N}_r$ of the component graph of $\mathcal{R}$.
19: Find decision sequences $D_1, \ldots, D_m$ and $A_1, \ldots, A_n$ relative to $\mathcal{D}$ and $\mathcal{A}$.
20: Apply Algorithm 3.
21: **End If**

The relevance graph may be built in quadratic time in the number of nodes in the BAID, and so may be found a topological ordering of the related component graph,

as claimed in Section 2.3.2. Therefore, Algorithm 4 runs in quadratic time in the number of nodes in the BAID as all processes involved, including Algorithms 2 and 3, have such computational complexity.

2.4 CIP: A numerical example

The proposed methodology is illustrated with the CIP example from Section 2.1. To support the defender, several assessments need to be obtained from her, regarding both her decision-making problem and her perspective on how the attacker would deal with his. Simplified assumptions shall be made to better depict the concepts. Then, Algorithm 4 is used to attain the defender's optimal decisions as well as the distribution over the attacker's random choices. In addition, a sensitivity analysis and some computational performance results are procured.

2.4.1 Assessing the defender's problem

To begin with, the defender's decisions are considered. First, she has to decide (D_1) whether to deploy additional measures to protect the infrastructure against terrorist attacks. Such choice is deemed to be binary: she reinforces the security ($d_1 = 1$) or she does not ($d_1 = 0$). After observing the outcome of the possible attack (S_1), she must determine (D_2) whether to implement recovery measures. Suppose that she has a specific budget to invest on relief and two options: mitigate the attack's consequences ($d_2 = 1$) or do nothing, saving the money ($d_2 = 0$).

Secondly, what the defender regards as the attacker's alternatives are dealt with. Both attacker's choices will be allowed to be binary. Thus, he must decide (A_1) about infiltrating one person within the infrastructure to gain intelligence for future attacks ($a_1 = 1$) or not ($a_1 = 0$). Then (A_2), he will determine whether to attack ($a_2 = 1$) or not ($a_2 = 0$).

The interaction between both agents' decisions results in two different random events. The first one is the outcome of the attack (S_1), dependent on D_1, A_1 and A_2, which shall be characterised through the number of days of service shortage ($s_1 \in \{0, \frac{1}{2}, 1\}$). Table 2.1 reflects the marginal distributions associated by the defender with each outcome s_1 depending on the combinations of d_1 and a_1 when there is an attack ($a_2 = 1$). For example, when the defender does not reinforce ($d_1 = 0$) and the attacker infiltrates the infrastructure ($a_1 = 1$), a half a day shortage ($s_1 = \frac{1}{2}$) has a probability 0.55 of occurrence. When there is no attack ($a_2 = 0$), all probability concentrates in no shortage days ($s_1 = 0$).

		(Reinforce, Infiltrate) - (d_1, a_1)			
		(0, 0)	**(0, 1)**	**(1, 0)**	**(1, 1)**
Shortage - s_1	**0**	0.30	0.15	0.40	0.25
	$\frac{1}{2}$	0.45	0.55	0.40	0.50
	1	0.25	0.30	0.20	0.25

Table 2.1: Defender's marginal distributions of S_1 given that $a_2 = 1$

The second random event (S_2) refers to the recovery impact, given D_2, A_2 and S_1. It will determine the number of days of service shortage reduction ($s_2 \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$), constrained by s_1. Table 2.2 presents the marginal distributions assessed by the defender about S_2 depending on s_1 when there is an attack ($a_2 = 1$) and the defender takes some action ($d_2 = 1$). For example, when facing a one day shortage ($s_1 = 1$), the probability of reducing it by one quarter ($s_2 = \frac{1}{4}$) is 0.15. Otherwise, when the attacker chooses not to attack ($a_2 = 0$) or the defender does not take any action ($d_2 = 0$), there is no shortage reduction ($s_2 = 0$).

		Shortage - s_1		
		0	**$\frac{1}{2}$**	**1**
Shortage Reduction - s_2	**0**	1.00	0.20	0.10
	$\frac{1}{4}$	0.00	0.50	0.15
	$\frac{1}{2}$	0.00	0.30	0.30
	$\frac{3}{4}$	0.00	0.00	0.25
	1	0.00	0.00	0.20

Table 2.2: Defender's marginal distributions of S_1 given that $a_2 = 1$

Last, the defender's utility $u_D(d_1, d_2, s_1, s_2)$ is assessed as suggested in González-Ortega, Radovic and Ríos Insua [2018]. A value function that effectively monetises all impacts is first considered: (i) security reinforcement ($d_1 = 1$) has a cost of $m_D^1 = 5$M€; (ii) the budget for relief ($d_2 = 1$) is of $m_D^2 = 10$M€; and (iii) each day of service shortage ($s_1 - s_2$) is considered to produce losses of $m_D^3 = 40$M€. The resulting value function (M€) is

$$v_D(d_1, d_2, s_1, s_2) = -m_D^1\, d_1 - m_D^2\, d_2 - m_D^3\, (s_1 - s_2).$$

Then, reckoning the defender as constant risk averse [Dyer and Sarin, 1982], the exponential utility function

$$u_D(d_1, d_2, s_1, s_2) = 1 - \exp(-\lambda_D \left(v_D(d_1, d_2, s_1, s_2) + c_D\right))$$

is determined, where $\lambda_D = 0.06$ is the risk aversion coefficient and $c_D = 55$ is an adjusting constant so that $0 \leq u_D \leq 1$.

2.4.2 Assessing the attacker's problem

The defender's perspective on the attacker's problem is now approached. Some of the assessments in her decision problem may be reused, such as the problem's structure or both agents' set of possible decisions. However, modelling the attacker's value and chance nodes is more involved as she is uncertain about his utility and probabilities.

Concerning the attacker's beliefs over the random events, the defender could base them on her own discrete probability distributions with some uncertainty around them by means of Dirichlet distributions as in Banks et al. [2015]. She would thus be assuming that the attacker's estimates are expected to agree with hers, but allowing for some uncertainty around them. This is just an operational heuristic and the attacker might have a completely different perception. If the situation were to be repeated over time, she could update the parameters in the Dirichlet distribution through Bayesian inference to obtain a posterior distribution that better reflects the attacker's estimates.

As an example, consider the marginal distribution of S_1 when the defender does not reinforce security ($d_1 = 0$), the attacker does not infiltrate the infrastructure ($a_1 = 0$) and there is an attack ($a_2 = 1$). The assessments for probabilities $q_i = p_D(s_1 = i \,|\, d_1 = 0, a_1 = 0, a_2 = 1)$ with $i = 0, \frac{1}{2}, 1$ in Table 2.1 are, respectively, 0.30, 0.45 and 0.25. However, the defender is uncertain about how the attacker would assess them; she associates random variables X_i such that $E[X_i] = q_i$ with $i = 0, \frac{1}{2}, 1$ and their joint distribution is Dirichlet, $(X_0, X_{1/2}, X_1) \sim \mathcal{D}ir(\alpha)$, with parameters $\alpha = (\alpha_0, \alpha_{1/2}, \alpha_1)$. If $\hat{\alpha} = \sum_i \alpha_i$, then:

$$E[X_i] = \frac{\alpha_i}{\hat{\alpha}}, \quad Var[X_i] = \frac{\alpha_i(\hat{\alpha} - \alpha_i)}{\hat{\alpha}^2(\hat{\alpha} + 1)}, \qquad i = 0, \tfrac{1}{2}, 1.$$

It suffices to fix a value for $Var[X_0]$ to calculate the required α_i parameters. If, for example, it is assumed that $Var[X_0] = (0.1 * E[X_0])^2 = (0.1 * 0.30)^2$, one gets $(X_0, X_{1/2}, X_1) \sim \mathcal{D}ir(69.70, 104.55, 58.08)$. Applying the same procedure to the other marginal distributions of the possible outcomes of S_1 for each combination of d_1 and a_1 when there is an attack ($a_2 = 1$), the Dirichlet distributions specified by their parameters in Table 2.3 are obtained. Again, when the attacker decides not to attack ($a_2 = 0$), there are no shortage days ($s_1 = 0$).

		(Reinforce, Infiltrate) - (d_1, a_1)			
		$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
Shortage - s_1	$\boldsymbol{\alpha_0}$	69.70	84.85	59.60	74.75
	$\boldsymbol{\alpha_{1/2}}$	104.55	311.12	59.60	149.50
	$\boldsymbol{\alpha_1}$	58.08	169.70	29.80	74.75

Table 2.3: Dirichlet α for the attacker's probabilities of S_1 given that $a_2 = 1$

For the second random event S_2, following a similar argument, Table 2.4 displays the parameters of the Dirichlet distributions when there is an attack ($a_2 = 1$) and the defender takes some action ($d_2 = 1$). Alternatively, there is no shortage reduction ($s_2 = 0$). Note that when the outcome S_2 is certain, the resulting Dirichlet distribution is degenerate; e.g. when there is no shortage ($s_1 = 0$).

		Shortage - s_1		
		$\mathbf{0}$	$\frac{\mathbf{1}}{\mathbf{2}}$	$\mathbf{1}$
Shortage Reduction - s_2	$\boldsymbol{\alpha_0}$	1.00	79.80	89.90
	$\boldsymbol{\alpha_{1/4}}$	0.00	199.50	134.85
	$\boldsymbol{\alpha_{1/2}}$	0.00	119.70	269.70
	$\boldsymbol{\alpha_{3/4}}$	0.00	0.00	224.75
	$\boldsymbol{\alpha_1}$	0.00	0.00	179.80

Table 2.4: Dirichlet α for the attacker's probabilities of S_2 given that $d_2, a_2 = 1$

$P_A(d_1)$ and $P_A(d_2 \,|\, d_1, a_2, s_1)$ are now assessed. To avoid recursions while acting on a level-2 thinking strategy, they are considered to be based on expert judgement allowing for some uncertainty. First, the defender has to choose D_1. Assume she concludes that the attacker's average probability for her to reinforce is 0.65. Then, she decides to use a Beta distribution $P_A(d_1 = 1) \sim \mathcal{B}e(13, 7)$ (and $P_A(d_1 = 0) = 1 - P_A(d_1 = 1)$). As with the Dirichlet distributions, Bayesian inference may be used to update the Beta distribution to better reflect the attacker's estimates.

Her second decision D_2 is more complex: she has to decide upon investing on recovery depending on an eventual attack (A_2), the outcome of that attack (S_1) and her reinforcing decision (D_1). Suppose the defender estimates the attacker's average

probabilities for her random (for him) decision D_2 as in Table 2.5. She may assume that the attacker would consider her recovery effort just dependent on the attack's outcome s_1, when there is an attack ($a_2 = 1$). For example, when confronting a one day shortage ($s_1 = 1$), the attacker's average probability for the defender to try to mitigate it ($d_2 = 1$) is 0.70. On the other hand, when the attacker decides not to attack ($a_2 = 0$), she may presume the attacker to expect her to do nothing ($d_2 = 0$) having no evidence upon which to act.

		Shortage - s_1		
		0	**$\frac{1}{2}$**	**1**
Recovery - d_2	**0**	1.00	0.60	0.30
	1	0.00	0.40	0.70

Table 2.5: Attacker's average probabilities of D_2 given that $a_2 = 1$

As before, the defender may introduce uncertainty in her assessment by means of Dirichlet distributions. Table 2.6 presents the matching parameters when there is an attack ($a_2 = 1$).

		Shortage - s_1		
		0	**$\frac{1}{2}$**	**1**
Recovery - d_2	**α_0**	1.00	39.40	69.70
	α_0	0.00	26.27	162.63

Table 2.6: Dirichlet α for the attacker's probabilities of D_2 given that $a_2 = 1$

Finally, the attacker's random utility $U_A(a_1, a_2, s_1, s_2)$ has to be determined. Following the guidelines in González-Ortega, Radovic and Ríos Insua [2018], suppose the defender has come up through expert judgement with an idea of the attacker's value function that relates all events in monetary terms: (i) infiltration ($a_1 = 1$) approximately costs him $m_A^1 = 1$M€; (ii) the attack ($a_2 = 1$) is expected to cost him $m_A^2 = 5$M€; and (iii) each day of service shortage ($s_1 - s_2$) is considered to profit him $m_A^3 = 35$M€. However, the defender is not sure about the precise parameters, so she could draw each of them uniformly over an interval of width 1M€ centred on the selected value; e.g. $M_A^1 \sim \mathcal{U}(0.5, 1.5)$ for the first parameter. Thus, the resulting random value function (M€) is

$$V_A(a_1, a_2, s_1, s_2) = -M_A^1 * a_1 - M_A^2 * a_2 + M_A^3 * (s_1 - s_2).$$

Then, she may assume that the attacker is constant risk prone [Dyer and Sarin, 1982] and use expert judgement to establish the exponential utility function

$$U_A(a_1, a_2, s_1, s_2) = \exp(\Lambda_A * (V_A(a_1, a_2, s_1, s_2) + C_A)),$$

where $\Lambda_A \sim \mathcal{U}(0.05, 0.07)$ represents the uncertain risk proneness coefficient and $C_A = -\max V_A$ is an adjusting constant so that $0 \leq U_A \leq 1$.

2.4.3 Supporting the defender

All the ingredients to support the defender are now available, while Algorithm 4 has been implemented in `R` (Appendix B.1). When using Monte Carlo simulation to approximate the defender's probabilities of the attacker's actions, $K = 10^5$ iterations have been used. The solution is presented in the manner of scheme $\mathcal{G}$ in Section 2.2:

$\mathcal{G}$1. Begin by solving decision D_2. Should there be no attack ($a_2 = 0$), there would be no consequences ($s_1 = 0$) and the defender would not have to mitigate them: $d_2^*(d_1, 0, 0) = 0$ with an expected utility of 0.963 (respectively, 0.950) if she had (respectively, had not) previously reinforced the system. However, if there is an attack ($a_2 = 1$), there are different possible scenarios. Table 2.7 displays the corresponding defender's optimal decisions and expected utilities: the optimal recovery decision is not to invest on relief ($d_2 = 0$) unless service shortage is most severe ($s_1 = 1$), when it is worth trying to mitigate it ($d_2 = 1$).

	(Reinforce, Shortage) - (d_1, s_1)					
	$\mathbf{(0,0)}$	$\mathbf{(0, \frac{1}{2})}$	$\mathbf{(0,1)}$	$\mathbf{(1,0)}$	$\mathbf{(1, \frac{1}{2})}$	$\mathbf{(1,1)}$
Recovery - d_2^*	0	0	1	0	0	1
Exp. Utility	0.963	0.878	0.754	0.950	0.835	0.668

Table 2.7: Defender's optimal decision $d_2^*(d_1, a_2, s_1)$ given that $a_2 = 1$

$\mathcal{G}$2. Next, deal with A_2. The probabilities $p_D(a_2 \,|\, d_1, a_1)$ in Table 2.8 have been obtained using Monte Carlo simulation. For example, when the defender does not reinforce security ($d_1 = 0$) and the attacker infiltrates the infrastructure ($a_1 = 1$), the attacker's probability of attack ($A_2^*(0, 1) = 1$) is 0.776. Note that it is always more likely that there is an attack ($a_2 = 1$), except when the defender reinforces security ($d_1 = 1$) and the attacker has not infiltrated anyone within the infrastructure ($a_1 = 0$).

		(Reinforce, Infiltrate) - (d_1, a_1)			
		(0, 0)	**(0, 1)**	**(1, 0)**	**(1, 1)**
Attack - A_2^*	**0**	0.339	0.224	0.593	0.430
	1	0.661	0.776	0.407	0.570

Table 2.8: Attacker's simulated probabilities for his optimal decision $A_2^*(d_1, a_1)$

$\mathcal{G}$3. The attacker's decision A_1 has to be now considered. Through Monte Carlo simulation one can determine that the probability of him infiltrating someone within the infrastructure ($A_1^* = 1$) is 0.282 and, thus, that the probability of him not doing so ($A_1^* = 0$) is 0.718.

$\mathcal{G}$4. Finally, solve the defender's decision D_1. Making use of the previous probabilities for A_1^* and $A_2^*(d_1, a_1)$, her optimal decision is to implement additional security measures ($d_1^* = 1$), expected utility 0.900; against not doing it ($d_1^* = 0$), expected utility 0.896.

Then, the defender's optimal strategy would be to reinforce the security ($d_1^* = 1$). Later, implement the mitigation policy ($d_2^* = 1$) in case of an attack ($a_2 = 1$) if the service shortage is maximum ($s_1 = 1$), and do nothing ($d_2^* = 0$) otherwise.

2.4.4 Sensitivity analysis

A solution to the CIP problem has been attained based on the proposed ARA approach. However, a more complete response would also provide some information about that solution's robustness. For this, a sensitivity analysis may be performed to check the impact of perturbations in the input. Robustness is specially important given the intrinsic uncertainty in the defender's assessment of the attacker's utility and probabilities.

To be concise, and as an example, just a brief sensitivity analysis about the parameters in the Beta distribution used to model $P_A(d_1)$ shall be performed. In Section 2.4.2, $P_A(d_1 = 1) \sim \mathcal{B}e(13, 7)$ (and $P_A(d_1 = 0) = 1 - P_A(d_1 = 1)$) was used. Table 2.9 displays the defender's optimal reinforcement decision d_1^* and its expected utility for slightly different choices of the parameters in the Beta distribution as well as, for comparison purposes, the estimated attacker's probability of infiltrating someone within the infrastructure, $p_D(a_1 = 1)$. Note that the defender's second decision D_2, related to the mitigation policy, would be unaffected by these parameter changes as D_1 is not strategically relevant for D_2.

Beta Parameters	$(11, 9)$	$(12, 8)$	$(13, 7)$	$(14, 6)$	$(15, 5)$
Avg. Pr. Reinforce	0.55	0.60	0.65	0.70	0.75
Reinforce - d_1^*	1	1	1	1	1
Exp. Utility	0.899	0.899	0.900	0.900	0.900
Pr. Infiltrate	0.293	0.288	0.282	0.279	0.276

Table 2.9: Sensitivity analysis of $P_A(d_1)$

It may be concluded that the solution is robust with respect to the initial estimation of $P_A(d_1)$, as the prescribed solution remains the same while its expected utility presents no significant variations. However, notice that the estimation of $p_D(a_1 = 1)$ does differ. When the attacker expects the defender to reinforce the infrastructure with a higher probability, he is further deterred to attack.

2.4.5 Computational performance

To conclude this section, the computation times of the implemented `R` routine for Algorithm 4 are provided considering different numbers of iterations for the Monte Carlo simulation to approximate the defender's probabilities of the attacker's actions. Simulations were run on a PC with the following specifications: *Operating System* - Windows 7 Professional 64 bits; *Processor* - Intel Core i5-6200 @ CPU 2.30 GHz (4 CPUs); *RAM Amount* - 8.00 GB; *Graphics* - Intel(R) HD Graphics 520. Table 2.10 presents the computation time for each number of iterations as well as the estimated attacker's probability of infiltrating someone within the infrastructure, $p_D(a_1 = 1)$, as a means of comparison.

Iterations - K	1	10	10^2	10^3	10^4	10^5
Comp. Time	0.14 sec	0.63 sec	5.62 sec	55.56 sec	9.23 min	1.53 h
Pr. Infiltrate	0.000	0.500	0.330	0.303	0.275	0.282

Table 2.10: Computation times for Algorithm 4

It may be observed that Algorithm 4 runs in linear time in the number of iterations used in the Monte Carlo simulation. This was to be expected, as the process of simulating the attacker's decisions is the most time consuming part of the algorithm.

In addition, note that the estimation of $p_D(a_1 = 1)$ becomes quite stable with a number of iterations (K) greater than 10^4.

2.5 Recapitulation

An ARA algorithmic framework to deal with proper BAIDs has been provided, thus avoiding strong common knowledge assumptions typical of earlier non-cooperative game-theoretic approaches to the problem. Within it, one of the agents is supported by essentially forecasting the actions of her adversary and then finding her optimal alternatives, which is better performed sequentially based on the informational needs of the sustained decision-maker exploiting the strategic relevance concept as in Koller and Milch [2003]. This leads to a procedure that combines simulation and optimisation stages switching between them according to the relevance and component graphs while identifying sequential and simultaneous decision sequences. The resulting algorithms employ a level-2 thinking strategy.

The methodology has been illustrated with a CIP example which could actually lead to a real model with some refinement, e.g. based on Zhang and Ramirez-Marquez [2013] or Bao et al. [2017]. However, the research has focused on the underlying interaction structure for which the example has just been illustrative and, therefore, on how to solve the model rather than to make it genuine and reliable.

As with standard IDs, the approach has quadratic complexity in the number of nodes. Besides, it has linear complexity in the number of iterations used for the Monte Carlo simulation adopted to estimate the distribution of the opponent's actions. To provide a solution to the numerical example, and as a pilot attempt, all algorithms have been implemented in `R` and are accessible in Appendix B.1.

Chapter 3

Concept uncertainty

The focus in this chapter is on concept uncertainty and how to handle it with an ARA approach. As explained in Section 1.4.2, concept uncertainty refers to the way in which the decision-maker believes her opponents frame the problem and choose their respective actions. To study it, the problem of Adversarial Statistical Decision Theory (ASDT) is introduced, extending the standard Statistical Decision Theory (SDT) formulation to consider an adversary who tries to modify the data-process observed by the decision-maker to confuse her and, consequently, attain a profit. Within the ASDT framework, the specific case of point estimation is used.

First, the idea of concept uncertainty is stressed with a qualitative security example based on spy drone detection in Section 3.1. Then, ASDT is presented in Section 3.2, while the methodology to cope with concept uncertainty through ARA is provided through the particular ASDT case of adversarial point estimation in Section 3.3. Finally, portraying the security example as an adversarial point estimation situation, a numerical example illustrates the approach in Section 3.4. A recap is included in Section 3.5.

3.1 Basic notions: A security example

The notion of concept uncertainty will be elaborated through a qualitative security example in relation to the detection of spy drones. Consider a defender (decision-maker) and an attacker (adversary) who make simultaneous decisions, with the defender being supported.

Although several methods have been proposed to detect nearby drones, among which Hearing and Franklin [2016], Shin et al. [2017] and Trundle and Slavin [2017] constitute a small representative collection, they all suffer from the same shortcoming:

they cannot determine what is being captured. Thus, they fail to discriminate between the legitimate use of a drone (e.g. for a home-made video) from an illegitimate use (e.g. to invade someone's privacy), as this distinction may rely on the video camera's orientation rather than on the drone's location. Nassi et al. [2018] suggest an external interception model using a radio frequency scanner to analyse the bit rate (number of bits conveyed per unit of time) of the live video-stream sent from the drone to the pilot (operator). In particular, they make use of the fact that changes in the number of pixels from a frame to its consecutive frame requires data to encode (intra-frame coding), therefore increasing the bit rate, and induce this to happen with physical stimuli to detect whether a drone is filming a certain target.

For the purpose of this example, assume the defender just wants to detect whether a drone is using its camera or not. Based on the method proposed by Nassi et al. [2018], assume she pretends to estimate the average data-stream bit rate of a certain drone model so as in the future be able to infer that she might be under surveillance when noticing it to be substantially higher. Suppose the attacker discovers this and may decide to perturb the process to try to be inconspicuous in case of spying the defender with that same kind of drone.

Typically, after both agents choose their actions, the outcomes and payoffs for each of them are random variables. In the example, regardless of the attacker's decision, there is a chance that the defender correctly evaluates the drone's average data-stream bit rate to detect its potential illicit use and a chance that she fails to determine it, providing a random amount of beneficial information to the attacker by snooping on the defender and causing a random amount of economic and/or personal damage to her. This randomness is aleatory uncertainty and is conditional on the choices of the opponents, yet it does not depend upon any strategic calculation on their parts, referring to the non-strategic randomness of an outcome. The defender should assess her beliefs about the outcome probabilities, conditional on the actions chosen by both agents. This can be addressed through traditional probabilistic risk analysis [Bedford and Cooke, 2001]. The defender's beliefs should be informed by expert judgement and previous history and extracted through appropriate elicitation methods as described by Cooke [1991] and O'Hagan et al. [2006], taking into account factors such as experts being over-confident and that previous history may be only partially relevant.

Epistemic uncertainty describes the defender's distributions over the choices the attacker will make, which usually integrate his preferences, beliefs and capabilities. For the spy drone detection example, the defender does not know how the attacker will affect the estimation process or whether he will actually do it. His choice would depend upon factors like which perturbation method is most valuable to him (a preference), the bit rate he thinks has the bigger chance of making him undetectable (a belief) and whether or not he has a clear way of influencing the procedure (a capability). The defender does not know these and, thus, has epistemic uncertainty. She expresses it as a distribution over all possible targets. This uncertainty

is handled differently for each solution concept that the defender thinks the attacker might use. For example, with a Nash equilibrium concept, the defender believes that the attacker thinks they both know the relevant utilities and beliefs; hence, the relevant epistemic uncertainty refers to the defender's distribution over the payoff bi-matrices that the attacker may be using. In the case of a Bayes-Nash equilibrium concept, there is additional uncertainty related to games of incomplete information [Harsanyi, 1967]. Besides the distribution over the entailed payoff bi-matrices, the defender must also express her epistemic uncertainty about the common knowledge distributions which the attacker is assuming that they share, their common distributions on types. In principle, full description of the epistemic uncertainties is complex, even with simple solution concepts. Often, there are pragmatic approximations that may be used.

Finally, as mentioned, concept uncertainty would arise from ignorance of how one's adversary will frame the analysis. In classical game theory terms, the defender does not know which solution concept the attacker will use to make his decision. Concept uncertainty embraces a wide range of strategies and is an essential component in the ARA formulation: the defender must model how the attacker will make his decision, but there are many possible solution concepts that he might use. Some of them are:

- *Non-strategic play.* The defender believes that the attacker will select an action without consideration of her choice. This includes the case in which the attacker selects actions with probabilities proportional to the perceived utility of success, Paté-Cornell and Guikema [2002]. It also allows for non-sentient opponents, such as a hurricane.

- *Nash or Bayes-Nash equilibrium methods.* The defender concludes that the attacker is assuming they both have a great deal of common knowledge and pursues an equilibrium point in which each agent's choice is the best response to their counterpart's action [Nash, 1951].

- *Level-k thinking.* The defender considers that the attacker thinks k steps deep in an "I think that she thinks that I think..." kind of reasoning as in Stahl and Wilson [1995]. Level-0 corresponds to non-strategic play.

- *Mirroring equilibrium analysis.* The defender supposes that the attacker is modelling her decision-making in the same way that she is modelling his, and both use subjective distributions on all unknown quantities [Banks et al., 2015].

As a Bayesian approach, ARA enables great flexibility in tailoring the analysis to the context and in accounting for the different kinds of uncertainty that arise. Usually, the defender does not know which solution concept the attacker has chosen. But based on previous experience with the attacker, and input from informants or other sources, she can place a subjective probability distribution over his possible

solution concepts. She could then make the decision that maximises her expected utility against that weighted mixture of strategies. Realistically, a full Bayesian analysis that puts positive probability on a large number of different solution concepts becomes computationally burdensome. However, in principle, the approach is simple. Each solution concept will lead (after handling the relevant epistemic and aleatory uncertainties) to a distribution over the attacker's actions. Then, the defender weights each distribution with her personal probability that the attacker is using such solution concept. This generates a weighted distribution on the attacker's action space which reflects all of the defender's knowledge about the problem and all of her uncertainty. The approach is related to Bayesian model averaging, in which uncertainty about the model is expressed by a probability distribution over the possible models and inference is based upon a weighted average of the posterior distributions from each model, see Hoeting et al. [1999] and Clyde and George [2004] for further details.

3.2 Adversarial statistical decision theory

As a motivation for the ASDT problem, a concise review of the standard Bayesian SDT framework is first provided. As depicted in the ID in Figure 3.1, a decision-maker needs to make a decision (D) based on an observation (X) depending on a state (Θ). She attains a loss $l_D(d, \theta)$ which is subject to her choice d and the state θ actually occurring.

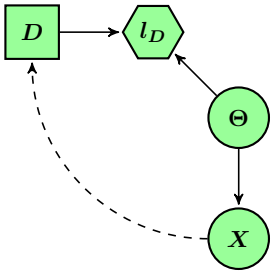


Figure 3.1: Sketch of the general SDT problem

To solve her decision-making problem, the decision-maker could describe her prior beliefs over the state θ through a prior $p_D(\theta)$ and the dependence of data x on the state θ through a likelihood $p_D(x \mid \theta)$. Given such elements, she would seek the decision $d^*(x)$ that minimises her posterior expected loss, given x, which is

$$d^*(x) = \arg\min_d \int l_D(d, \theta)\, p_D(\theta \mid x)\, \mathrm{d}\theta. \tag{3.1}$$

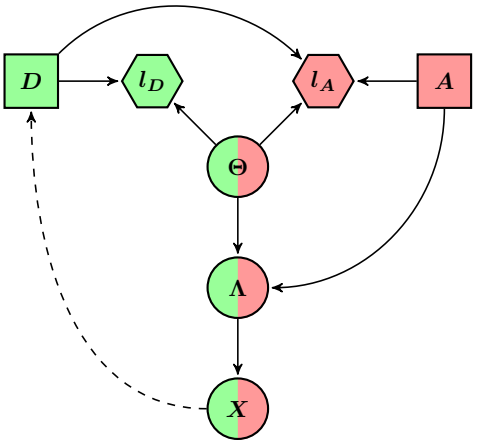
Note that, for optimization purposes, the denominator $p_D(x)$ in Bayes' formula may

be ignored and problem

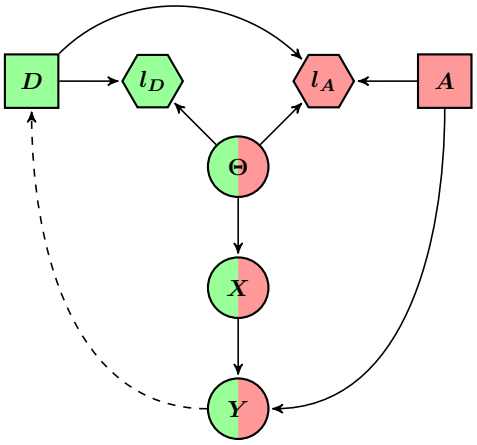
$$d^*(x) = \arg\min_d \int l_D(d,\theta)\, p_D(x \,|\, \theta)\, p_D(\theta)\, \mathrm{d}\theta \tag{3.2}$$

be solved, which is equivalent to (3.1). This transformation allows to directly involve the probabilities $p_D(\theta)$ and $p_D(x \,|\, \theta)$ and, in principle, avoid computing the more complex posterior $p_D(\theta \,|\, x)$. By appropriately crafting the decision space and the loss function, this framework embraces most usual statistical problems including point estimation, set estimation, hypothesis testing, forecasting and decision analysis, as reviewed in detail in e.g. French and Ríos Insua [2000].

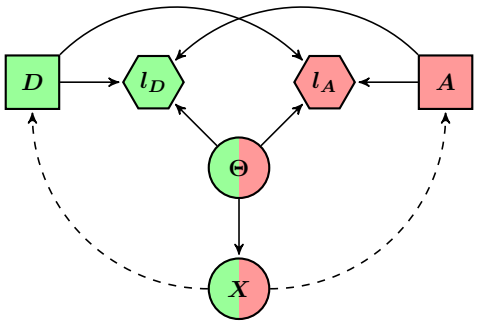
There are many possible variants of the SDT framework that extend it to the ASDT problem by considering a strategic adversary who makes decisions in the previous context which affect the information and/or consequences that the decision-maker obtains to acquire some advantage. Among all several adversarial scenarios that may actually be considered, three of the simpler ones are described through their BAIDs in Figure 3.2:



(a) Structural adversary (S1)



(b) Data-fiddler adversary (S2)



(c) Simultaneous adversary (S3)

Figure 3.2: BAIDs for three ASDT scenarios

S1. The adversary somehow manages to transform state θ affecting the data-process leading to a modified one $a(\theta) = \lambda$. This may be called a *structural adversary*, shown in Figure 3.2a.

S2. The adversary manages to modify the data x to be received by the decision-maker, who actually observes $a(x) = y$. This may be designated a *data-fiddler adversary*, depicted in Figure 3.2b, which reflects the typical case in adversarial machine learning, see e.g. Dalvi et al. [2004] or Tygar [2011].

S3. The adversary makes decisions a so that the losses for each agent, which are respectively denoted $l_D(d, \theta, a)$ and $l_A(d, \theta, a)$, depend upon both their choices and the relevant state. Other than that, the opponent faces a problem structurally similar to that of the decision-maker. This may be named a *simultaneous adversary*, represented in Figure 3.2c.

Note that the three formulations could appear combined in certain scenarios. For example, in the spy drone detection problem, during the defender's estimation of the average data-stream bit rate the attacker might artificially decrease the bit rate (θ); perturb the Wi-Fi signal that her radio frequency scanner captures (x); and, in addition, undertake adaptive modifications on his own drone (a), contesting the defender's detection strategy.

3.3 Adversarial point estimation

The relevance of concept uncertainty is now studied showcasing it in the standard problem of point estimation, described in Lehmann and Casella [1983]. The problem is first understood as a specific case of SDT and later adapted to an adversarial situation by acknowledging the presence of an adversary willing to deceive the supported decision-maker, in line with the ASDT framework. Lastly, concept uncertainty is properly engaged.

3.3.1 Point estimation as a SDT problem

The SDT problem reviewed in the previous section may be used to reflect a point estimation problem. Just as illustrated in Figure 3.1, a decision-maker is required to determine a value $d \in \Theta$ after observing x which depends on a state θ taking values in the same set Θ, obtaining a loss $l_D(d, \theta)$. Her optimal decision would then be found by minimising (3.1) or, more appropriately, by using its equivalent version (3.2).

As an example, under the quadratic loss function $l_D(d,\theta) = (\theta - d)^2$, it may be easily obtained that $d^*(x) = E[\theta \mid x]$ from (3.1) [French and Ríos Insua, 2000]. The optimisation argument with the quadratic loss is next rehearsed in the simplified form (3.2). The objective function to be minimised in this case is

$$\int \theta^2 \, p_D(x \mid \theta) \, p_D(\theta) \, \mathrm{d}\theta + d^2 \int p_D(x \mid \theta) \, p_D(\theta) \, \mathrm{d}\theta - 2d \int \theta \, p_D(x \mid \theta) \, p_D(\theta) \, \mathrm{d}\theta,$$

which is equivalent to minimising in d

$$d^2 \, p_D(x) - 2d \int \theta \, p_D(x \mid \theta) \, p_D(\theta) \, \mathrm{d}\theta,$$

whose minimum is, as announced,

$$d^*(x) = \frac{1}{p_D(x)} \int \theta \, p_D(x \mid \theta) \, p_D(\theta) \, \mathrm{d}\theta = \int \theta \, p_D(\theta \mid x) \, \mathrm{d}\theta = E[\theta \mid x]. \tag{3.3}$$

3.3.2 Adversarial point estimation as an ASDT problem

To streamline the discussion, only the ASDT Scenario S1 in which an structural adversary is capable of perturbing θ (Figure 3.2a) will be considered. Specifically, suppose that the opponent just uses deterministic transformations $\lambda = \theta + a$, biasing the underlying state, with a chosen by him. In addition, a quadratic loss function for the decision-maker will be assumed. Note first that a decision-maker who is not aware of the presence of the adversary would be proposing as optimal the decision in (3.2), without noticing that $p_D(x \mid \theta)$ should be updated to $p_D(x \mid \theta + a)$ as the data-process is being perturbed and, therefore, systematically erring her estimation in general.

Consider now an adversary-aware point estimator. She faces a problem similar to that in Figure 3.1, except for the additional uncertainty about a, as reflected in the ID in Figure 3.3a, which modifies the state θ. Recall that as the decision-maker does not know her adversary's choice, his corresponding decision node (A) appears as random to her. Once she makes a forecast $p_D(a)$ of the action a to be performed by her opponent, the problem that she needs to solve is formulated (equivalently) as

$$d^*(x) = \arg\min_d \iiint (\theta - d)^2 \, p_D(x \mid \lambda) \, p_D(\lambda \mid \theta, a) \, p_D(\theta) \, p_D(a) \, \mathrm{d}\lambda \, \mathrm{d}\theta \, \mathrm{d}a. \tag{3.4}$$

In the same manner as the non-adversarial version in Section 3.3.1, observe that this leads to the equivalent problem of minimising in d

$$d^2 \, p_D(x) - 2d \iiint \theta \, p_D(x \mid \lambda) \, p_D(\lambda \mid \theta, a) \, p_D(\theta) \, p_D(a) \, \mathrm{d}\lambda \, \mathrm{d}\theta \, \mathrm{d}a,$$

whose solution is

$$d^*(x) = \frac{1}{p_D(x)} \iiint \theta \, p_D(x \,|\, \lambda) \, p_D(\lambda \,|\, \theta, a) \, p_D(\theta) \, p_D(a) \, \mathrm{d}\lambda \, \mathrm{d}\theta \, \mathrm{d}a.$$

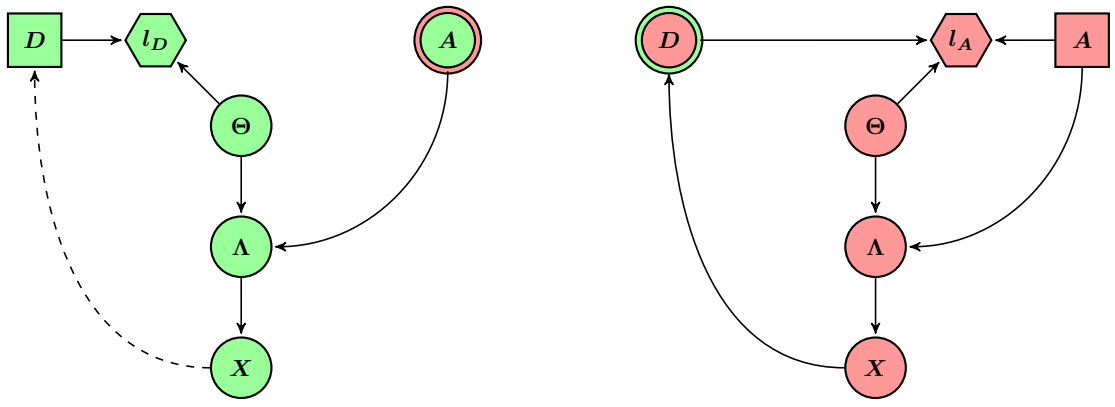
This expression may then be simplified to obtain

$$\begin{aligned} d^*(x) &= \frac{1}{p_D(x)} \iint \theta \, p_D(x \,|\, \lambda) \, p_D(\lambda \,|\, \theta) \, p_D(\theta) \, \mathrm{d}\lambda \, \mathrm{d}\theta \\ &= \frac{1}{p_D(x)} \int \theta \, p_D(x \,|\, \theta) \, p_D(\theta) \, \mathrm{d}\theta = \int \theta \, p_D(\theta \,|\, x) \, \mathrm{d}\theta = E\left[\theta \,|\, x\right]. \end{aligned}$$

Apparently, the same solution as in (3.3) is reached. However, these compressed expressions do not explicitly show that the probability model $p_D(\theta \,|\, x)$ is different in both cases. Therefore, it is convenient to use decomposed expressions such as (3.4) which involve probabilities that are easier to model and learn from, in line with the Fermitisation point of view mentioned in Section 1.4.2. Due to the deterministic transformation assumption, in this specific case $p_D(\lambda = \theta + a \,|\, \theta, a) = 1$ (and 0 otherwise). Thus, (3.4) becomes

$$d^*(x) = \arg\min_d \iint (\theta - d)^2 \, p_D(x \,|\, \lambda = \theta + a) \, p_D(\theta) \, p_D(a) \, \mathrm{d}\theta \, \mathrm{d}a. \tag{3.5}$$

It shall be remarked once again that a good assessment of $p_D(a)$ is therefore crucial, although it is complex because of the strategic element involved. To better assess it, the decision-maker may consider her adversary's problem reflected in the ID in Figure 3.3b, as conducted in Section 2.1, with her decision (D) being a chance node to him. Concept uncertainty needs to be faced at this point.



(a) Decision-maker's problem

(b) Adversary's problem

Figure 3.3: Both agents' IDs for a structural adversary in ASDT

3.3.3 Addressing concept uncertainty

For the sake of simplicity, just two different solution concepts for the adversary (Bayesian and minimax) will be proposed. A way to aggregate them in terms of the defender's uncertainty will be explained.

A Bayesian adversary

Suppose first that the adversary minimises expected loss. He would seek to find the optimal value a to transform state θ such that, when the decision-maker observes x and estimates d, his expected loss $l_A(d, \theta, a)$ is minimum. To solve his problem, the opponent would need to express his prior beliefs over the state θ through a prior $p_A(\theta)$ and the dependence of data x on the state θ and the defender's choice d on observation x, respectively, through likelihoods $p_A(x \mid \lambda = \theta + a)$ and $p_A(d \mid x)$. Then, his optimal decision would be computed as

$$a_B^* = \arg\min_a \iiint l_A(d, \theta, a)\, p_A(d \mid x)\, p_A(x \mid \lambda = \theta + a)\, p_A(\theta)\, \mathrm{d}d\, \mathrm{d}x\, \mathrm{d}\theta.$$

However, the decision-maker does not know her adversary's loss function and probabilities. If she acknowledges her uncertainty about them through random losses and probabilities $F \sim \big(L_A(d, \theta, a), P_A(d \mid x), P_A(x \mid \lambda = \theta + a), P_A(\theta)\big)$, she would solve

$$A_B^* = \arg\min_a \iiint L_A(d, \theta, a)\, P_A(d \mid x)\, P_A(x \mid \lambda = \theta + a)\, P_A(\theta)\, \mathrm{d}d\, \mathrm{d}x\, \mathrm{d}\theta$$

to find the optimal random decision A_B^*, which is a random variable whose distribution is induced by the above random loss function and probabilities. Then, as in Section 2.2, the decision-maker would have found the distribution that she needs to calculate her best decision $d^*(x)$. That distribution would incorporate all of her uncertainty about her adversary's decision-making context, assuming that he is Bayesian, and may be defined through $p_D^B(a) = \Pr(A_B^* = a)$ in the discrete case and, similarly, in the continuous one.

To approximate $p_D^B(a)$, one would typically use Monte Carlo simulation, drawing K samples $\big(L_A^k(d, \theta, a), P_A^k(d \mid x), P_A^k(x \mid \lambda = \theta + a), P_A^k(\theta)\big)$, $k = 1, \ldots, K$ from F, finding

$$A_{B,k}^* = \arg\min_a \iiint L_A^k(d, \theta, a)\, P_A^k(d \mid x)\, P_A^k(x \mid \lambda = \theta + a)\, P_A^k(\theta)\, \mathrm{d}d\, \mathrm{d}x\, \mathrm{d}\theta \qquad (3.6)$$

and approximating

$$\hat{p}_D^B(a) \approx \#\{A_{B,k}^* = a\}/K.$$

Within F, $P_A(d \mid x)$ is much harder to assess than the other three elements. It entails strategic thinking, since the decision-maker needs to understand her adversary's

beliefs on what point estimates she will make given that she observes x. This could be the prelude of a hierarchy of decision-making problems if the decision-maker assumes that her opponent regards her as adversary-aware; see Ríos and Ríos Insua [2012] for a description of the potentially infinite regress in a simpler class of problems. For illustrative purposes, the immediate stage of the hierarchy in this case is presented. Note that in the problem

$$d^*(x) = \arg\min_d \iiint l_D(d,\theta)\, p_D(x \mid \lambda)\, p_D(\lambda \mid \theta, a)\, p_D(\theta)\, p_D(a)\, \mathrm{d}\lambda\, \mathrm{d}\theta\, \mathrm{d}a$$

to be solved by the decision-maker, her adversary lacks knowledge about the terms in the integral. Assuming uncertainty about them through a random loss $L_D^A(d,\theta)$ and random distributions $P_D^A(x \mid \lambda)$, $P_D^A(\lambda \mid \theta, a)$, $P_D^A(\theta)$ and $P_D^A(a)$, he would get the corresponding random optimal decision by replacing the matching elements. Again, this demands assessment of $P_D^A(a)$ (what the decision-maker believes that her adversary thinks about her beliefs concerning his action to be implemented) for which there is a strategic component, leading to the next stage in the hierarchy. In line with the already introduced level-k thinking strategies, one could stop at a pertinent level in which no more information is reasonably available. At that point, one could use a non-informative prior over the involved losses and probabilities.

A minimax adversary

Assume now that rather than minimising posterior expected loss, the decision-maker feels that her adversary behaves in a minimax manner. This means that he would try to minimise his possible maximum loss through

$$a_M^* = \arg\min_a \max_{d,\theta} l_A(d,\theta,a).$$

However, as in Section 3.3.3, the decision-maker does not know her opponent's loss function. If she ascertains some uncertainty about it through a random loss $L_A(d,\theta,a)$, she would solve

$$A_M^* = \arg\min_a \max_{d,\theta} L_A(d,\theta,a),$$

to find the optimal random decision A_M^*. Again, the distribution of A_M^* incorporates all of her uncertainty about her adversary's decision-making context, suspecting this time that he is a minimax adversary. Thus, it may be defined as $p_D^M(a) = \Pr(A_M^* = a)$ in the discrete case and, similarly, in the continuous one.

Monte Carlo simulation would typically be used to estimate $p_D^M(a)$, as in Section 3.3.3, drawing K samples from $L_A^k(d,\theta,a)$, $k = 1,\ldots,K$, and computing

$$A_{M,k}^* = \arg\min_a \max_{d,\theta} L_A^k(d,\theta,a),$$

which would lead to

$$\hat{p}_D^M(a) \approx \#\{A_{M,k}^* = a\}/K.$$

Aggregating solution concepts

The above adversary types could be extended with other solution concepts, but for the purpose of this section two of them are enough. If the decision-maker believes that her adversary is Bayesian with weight π_B and minimax with weight π_M, with $\pi_B + \pi_M = 1$ and $\pi_B, \pi_M \geq 0$, then the required forecast $p_D(a)$ would be based on $\pi_B \hat{p}_D^B(a) + \pi_M \hat{p}_D^M(a)$ which would be plugged in her decision-making problem (3.5), leading to an optimal adversary-aware point estimation.

3.4 Spy drone detection: A numerical example

Coming back to the spy drone detection setting from Section 3.1, a numerical example is included to make the ARA approach to concept uncertainty clear. Note first that, essentially, the defender faces an adversarial point estimation process in which she tries to assess a drone's average data-stream bit rate while the attacker might perturb the procedure. Assuming the attacker is an structural adversary and may do this by artificially decreasing the bit rate, the same conditions as in Section 3.3.3 are met. Consider that the defender believes that:

- Drones' average data-stream bit rates follow a normal prior $\theta \sim \mathcal{N}(\mu_D, \sigma_D)$. In particular, $\mu_D = 200$ Kbps and $\sigma_D = 25$ Kbps shall be set.
- Observations $x = (x_1, \ldots, x_n)$ will be i.i.d. around bit rate θ as $x_i \,|\, \theta \sim \mathcal{N}(\theta, \rho_D)$ for $i = 1, \ldots, n$. Specifically, $\rho_D = 10$ Kbps shall be stipulated.

When the defender is unaware of any artificial decrease of the data-stream bit rate made by the attacker, her optimal estimation of the average bit rate may be computed by means of (3.2), given observations $(x_1, \ldots, x_n)$, minimising in d

$$\int (\theta - d)^2 \, \frac{1}{\sigma_D \sqrt{2\pi}} \, \exp\left(\frac{-(\theta - \mu_D)^2}{2\sigma_D^2}\right) \prod_{i=1}^{n} \frac{1}{\rho_D \sqrt{2\pi}} \, \exp\left(\frac{-(x_i - \theta)^2}{2\rho_D^2}\right) \mathrm{d}\theta$$

$$\propto$$

$$\int (\theta - d)^2 \, \frac{\sqrt{\rho_D^2 + n\,\sigma_D^2}}{\rho_D \, \sigma_D \sqrt{2\pi}} \, \exp\left(\frac{-\left(\theta - \frac{\mu_D \, \rho_D^2 + \sigma_D^2 \sum_{i=1}^{n} x_i}{\rho_D^2 + n\,\sigma_D^2}\right)^2}{2\,\frac{\rho_D^2 \, \sigma_D^2}{\rho_D^2 + n\,\sigma_D^2}}\right) \mathrm{d}\theta,$$

which is equivalent to minimising in d

$$d^2 \int \frac{\sqrt{\rho_D^2+n\,\sigma_D^2}}{\rho_D\,\sigma_D\sqrt{2\pi}} \exp\left(\frac{-\left(\theta-\frac{\mu_D\,\rho_D^2+\sigma_D^2\sum_{i=1}^n x_i}{\rho_D^2+n\,\sigma_D^2}\right)^2}{2\,\frac{\rho_D^2\,\sigma_D^2}{\rho_D^2+n\,\sigma_D^2}}\right)\mathrm{d}\theta$$

$$-\,2d\int \theta\,\frac{\sqrt{\rho_D^2+n\,\sigma_D^2}}{\rho_D\,\sigma_D\sqrt{2\pi}} \exp\left(\frac{-\left(\theta-\frac{\mu_D\,\rho_D^2+\sigma_D^2\sum_{i=1}^n x_i}{\rho_D^2+n\,\sigma_D^2}\right)^2}{2\,\frac{\rho_D^2\,\sigma_D^2}{\rho_D^2+n\,\sigma_D^2}}\right)\mathrm{d}\theta.$$

Both integrals include the density function of the normal distribution

$$Z \sim \mathcal{N}\left(\frac{\mu_D\,\rho_D^2+\sigma_D^2\sum_{i=1}^n x_i}{\rho_D^2+n\,\sigma_D^2}, \frac{\rho_D\,\sigma_D}{\sqrt{\rho_D^2+n\,\sigma_D^2}}\right)$$

so the problem may be simplified to minimising in d

$$d^2 - 2d\,E\,[Z] = d^2 - 2d\,\frac{\mu_D\,\rho_D^2+\sigma_D^2\sum_{i=1}^n x_i}{\rho_D^2+n\,\sigma_D^2},$$

whose optimal value is the mean in the Bayesian posterior normal distribution with known variance

$$d^*(x) = \frac{\mu_D\,\rho_D^2+\sigma_D^2\sum_{i=1}^n x_i}{\rho_D^2+n\,\sigma_D^2} = \frac{1}{\frac{1}{\sigma_D^2}+\frac{n}{\rho_D^2}}\left(\frac{\mu_D}{\sigma_D^2}+\frac{\sum_{i=1}^n x_i}{\rho_D^2}\right). \tag{3.7}$$

In particular, with the above specific parameters:

$$d^*(x) = \frac{32+\sum_{i=1}^n x_i}{0.16+n}.$$

Suppose now that the defender is aware of the attacker's presence. Then, she needs to further model her perspective on the attacker's losses and probabilities. Assume that the defender thinks the attacker:

- Considers drones' average data-stream bit rates follow a normal distribution $\theta \sim \mathcal{N}(\mu_A, \sigma_A)$. Moreover, $\mu_A \sim \mathcal{U}(175, 225)$ and $\sigma_A \sim \mathcal{U}(20, 30)$, in Kbps, reflect her uncertainty about the actual parameters he uses.
- Is restricted to just two actions, $a \in \{\text{-}25, 0\}$, related with decreasing the bit rate in 25 units (a = -25 Kbps, with the bit rate becoming $\lambda = \theta - 25$) or doing nothing ($a = 0$ Kbps, with the bit rate unperturbed as $\lambda = \theta$).
- Takes the observations $(x_1, \ldots, x_n)$ to be i.i.d. and normal around the transformed bit rate λ as $x_i \,|\, \lambda \sim \mathcal{N}(\lambda, \rho_A)$ for $i = 1, \ldots, n$ where $\rho_A \sim \mathcal{U}(5, 15)$, in Kbps, models her uncertainty.

- Is a Bayesian adversary with probability $\pi_B = \frac{2}{3}$ and a minimax adversary with probability $\pi_M = \frac{1}{3}$.

- Adopts a loss function that incorporates a penalty in relation with a and a quadratic term regarding her estimation d, so that $L_A(d, \theta, a) = \alpha\,|a| - \beta\,(\theta - d)^2$. Note that the attacker wins, on one hand, when the defender's estimate is further distant from θ, and, on the other hand, as his perturbation is smaller. We model the defender's uncertainty about the loss parameters with $\alpha \sim \mathcal{U}(3, 5)$ and $\beta \sim \mathcal{U}(0.1, 0.3)$.

- Determines that she will make her decision (point estimate of θ) uniformly around $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ with $d \,|\, x \sim \mathcal{U}(\bar{x} - 10, \bar{x} + 10)$, in Kbps.

Note first that when the attacker is a minimax adversary as in Section 3.3.3, his maximum loss always corresponds with the defender correctly guessing the average data-stream bit rate θ aggravated by his choice of a. Therefore, he would be compelled to do nothing ($a = 0$ Kbps) and, thus:

$$\hat{p}_D^M(\text{-}25) = 0, \qquad \hat{p}_D^M(0) = 1.$$

This leads to the same decision (3.7) as in the non-adversarial problem, as can be verified by replacing $p_D(a)$ with $\hat{p}_D^M(a)$ in (3.5).

On the contrary, when the attacker is Bayesian as in Section 3.3.3, simulation scheme (3.6) may be applied. To keep computations simple, the attacker will be presumed to consider that the defender will just make $n = 1$ observations, though uncertainty around this fact could also be introduced. A computational environment implementing (3.6) has been developed in `MATLAB` (Appendix B.2). Using $K = 10^4$ iterations, the following probabilities are obtained:

$$\hat{p}_D^B(\text{-}25) = 0.695, \qquad \hat{p}_D^B(0) = 0.305.$$

Thus, he is more likely to perform his attack than not. Then, making use of (3.5), the defender may compute her optimal decision, given observations $(x_1, \ldots, x_n)$, by minimising in d

$$\hat{p}_D^B(\text{-}25) \int (\theta - d)^2 \, \frac{1}{\sigma_D\sqrt{2\pi}} \exp\left(\frac{-(\theta-\mu_D)^2}{2\sigma_D^2}\right) \prod_{i=1}^{n} \frac{1}{\rho_D\sqrt{2\pi}} \exp\left(\frac{-(x_i-\theta+25)^2}{2\rho_D^2}\right) \mathrm{d}\theta$$

$$+\, \hat{p}_D^B(0) \int (\theta - d)^2 \, \frac{1}{\sigma_D\sqrt{2\pi}} \exp\left(\frac{-(\theta-\mu_D)^2}{2\sigma_D^2}\right) \prod_{i=1}^{n} \frac{1}{\rho_D\sqrt{2\pi}} \exp\left(\frac{-(x_i-\theta)^2}{2\rho_D^2}\right) \mathrm{d}\theta.$$

This may be simplified in the same manner as the non-adversarial version to find

the equivalent problem of minimising in d

$$\hat{p}_D^B(\text{-}25)\,\xi(x,\text{-}25)\left(d^2 - 2d\,\frac{\mu_D\,\rho_D^2+\sigma_D^2\sum_{i=1}^n (x_i+25)}{\rho_D^2+n\,\sigma_D^2}\right)$$

$$+\,\hat{p}_D^B(0)\,\xi(x,0)\left(d^2 - 2d\,\frac{\mu_D\,\rho_D^2+\sigma_D^2\sum_{i=1}^n x_i}{\rho_D^2+n\,\sigma_D^2}\right)$$

where

$$\xi(x,a) = \exp\left(\frac{\left(\mu_D\,\rho_D^2+\sigma_D^2\sum_{i=1}^n (x_i-a)\right)^2}{2\,\rho_D^2\,\sigma_D^2\left(\rho_D^2+n\,\sigma_D^2\right)} - \frac{\sum_{i=1}^n (x_i-a)^2}{2\,\rho_D^2}\right),$$

whose minimum is

$$d^*(x) = \frac{\mu_D\,\rho_D^2}{\rho_D^2+n\,\sigma_D^2}$$

$$+\,\sigma_D^2\,\frac{\hat{p}_D^B(\text{-}25)\,\xi(x,\text{-}25)\sum_{i=1}^n (x_i+25) + \hat{p}_D^B(0)\,\xi(x,0)\sum_{i=1}^n x_i}{\left(\rho_D^2+n\,\sigma_D^2\right)\left(\hat{p}_D^B(\text{-}25)\,\xi(x,\text{-}25)+\hat{p}_D^B(0)\,\xi(x,0)\right)}.$$

Employing the explicit parameters:

$$d^*(x) = \frac{32 + \dfrac{0.695\,\xi(x,\text{-}25)\sum_{i=1}^n (x_i+25) + 0.305\,\xi(x,0)\sum_{i=1}^n x_i}{0.695\,\xi(x,\text{-}25)+0.305\,\xi(x,0)}}{0.16+n}$$

with

$$\xi(x,a) = \exp\left(\frac{\left(32+\sum_{i=1}^n (x_i-a)\right)^2}{32+200n} - \frac{\sum_{i=1}^n (x_i-a)^2}{200}\right).$$

Aggregating the different solution concepts (Bayesian and minimax) as in Section 3.3.3, the defender's optimal decision may be determined, given observations $(x_1,\ldots,x_n)$, by minimising in d

$$\pi_B\,\hat{p}_D^B(\text{-}25)\int (\theta-d)^2\,\frac{1}{\sigma_D\sqrt{2\pi}}\,\exp\left(\frac{-(\theta-\mu_D)^2}{2\sigma_D^2}\right)\prod_{i=1}^n \frac{1}{\rho_D\sqrt{2\pi}}\,\exp\left(\frac{-(x_i-\theta+25)^2}{2\rho_D^2}\right)\mathrm{d}\theta$$

$$+\left(\pi_M+\pi_B\,\hat{p}_D^B(0)\right)\int (\theta-d)^2\,\frac{1}{\sigma_D\sqrt{2\pi}}\,\exp\left(\frac{-(\theta-\mu_D)^2}{2\sigma_D^2}\right)\prod_{i=1}^n \frac{1}{\rho_D\sqrt{2\pi}}\,\exp\left(\frac{-(x_i-\theta)^2}{2\rho_D^2}\right)\mathrm{d}\theta,$$

whose solution is

$$d^*(x) = \frac{\mu_D\,\rho_D^2}{\rho_D^2+n\,\sigma_D^2}$$

$$+\,\sigma_D^2\,\frac{\pi_B\,\hat{p}_D^B(\text{-}25)\,\xi(x,\text{-}25)\sum_{i=1}^n (x_i+25) + \left(\pi_M+\pi_B\,\hat{p}_D^B(0)\right)\,\xi(x,0)\sum_{i=1}^n x_i}{\left(\rho_D^2+n\,\sigma_D^2\right)\left(\pi_B\,\hat{p}_D^B(\text{-}25)\,\xi(x,\text{-}25)+\left(\pi_M+\pi_B\,\hat{p}_D^B(0)\right)\xi(x,0)\right)}.$$

Then, using the specific parameters:

$$d^*(x) = \frac{32 + \dfrac{0.436\,\xi(x,\text{-}25)\sum_{i=1}^{n}(x_i+25) + 0.537\,\xi(x,0)\sum_{i=1}^{n} x_i}{0.436\,\xi(x,\text{-}25) + 0.537\,\xi(x,0)}}{0.16+n}.$$

Table 3.1 summarises the different optimal estimations obtained with all the considered solution concepts. As stressed in Section 3.3.2, in an adversary-unaware situation (non-adversarial) the defender would systematically misestimate the drone's average data-stream bit rate in case that the attacker perturbs the process. With an ARA approach, the defender would fully acknowledge his presence in the problem and take into account all reasonable attack strategies through concept uncertainty, adjusting her assessment accordingly.

Solution Concept	**Optimal Estimation**
Non-Adversarial	$\dfrac{32+\sum_{i=1}^{n} x_i}{0.16+n}$
Minimax Adv.	$\dfrac{32+\sum_{i=1}^{n} x_i}{0.16+n}$
Bayesian Adv.	$\dfrac{32 + \dfrac{0.695\,\xi(x,\text{-}25)\sum_{i=1}^{n}(x_i+25) + 0.305\,\xi(x,0)\sum_{i=1}^{n} x_i}{0.695\,\xi(x,\text{-}25) + 0.305\,\xi(x,0)}}{0.16+n}$
Uncertain Adv.	$\dfrac{32 + \dfrac{0.436\,\xi(x,\text{-}25)\sum_{i=1}^{n}(x_i+25) + 0.537\,\xi(x,0)\sum_{i=1}^{n} x_i}{0.436\,\xi(x,\text{-}25) + 0.537\,\xi(x,0)}}{0.16+n}$

Table 3.1: Defender's optimal estimations for each solution concept

3.5 Recapitulation

Concept uncertainty within the ARA framework has been exemplified in ASDT, specified through point estimation. Only one out of the three outlined ASDT formulations, that of structural adversaries, was dwelt with using concept uncertainty, yet that of data-fiddler adversaries is deeply studied under an ARA perspective portrayed as the AHT problem in next chapter. The approach has been general extending beyond point estimation and may be applied to other standard statistical problems, including not only hypothesis testing but also interval estimation and

forecasting. Two solution concepts have been used, namely Bayesian and minimax adversaries, along with an aggregating procedure to deal with concept uncertainty.

A spy drone detection example based on Nassi et al. [2018] has illustrated an adversarial point estimation situation. This could be further sophisticated by means of considering the adversary to be not just structural but also data-fiddler and/or simultaneous. The expressions providing the optimal estimations have been numerically approximated in `MATLAB`, with code available in Appendix B.2.

Chapter 4

Adversarial hypothesis testing

In the present chapter, an alternative novel ARA approach to the AHT problem is procured. This problem considers a decision-maker who needs to assess which of several hypotheses holds, based on observations from a source that might have been disturbed by an adversary. In line with Dalvi et al. [2004] and Barni and Tondi [2014], the adopted basic AHT framework coincides with that of ASDT Scenario S2 in Section 3.2 in which the opponent is regarded as a data-fiddler adversary. However, a modification of the problem's structure is also developed, exemplified in a batch acceptance context, to reflect an alternative situation where the decision-maker only has partial information on the observations.

To start with, Section 4.1 devises the ASDT problem under the case of a data-fiddler adversary. In Section 4.2, the elementary AHT problem under an ARA perspective is formalised and a conceptual solution focusing on binary point hypothesis testing supplied, as well as an illustrative numerical example and a comparison with a game-theoretic paradigm. Last, Section 4.3 describes in depth an extension of the above AHT model to an application associated with batch acceptance. Section 4.4 provides a recapitulation.

4.1 Data-fiddler adversaries in ASDT

The problem of ASDT considering data-fiddler adversaries was illustrated in Figure 3.2b of the previous chapter through a BAID. Essentially, depending on an uncertain state θ, a decision-maker ought to receive some original data x and then make a decision d for which she perceives a loss $l_D(d, \theta)$. However, x gets perturbed in advance into y through her adversary's action a who obtains a loss $l_A(d, \theta, a)$. Thus, the decision-maker actually observes y, while x and θ remain unknown to her.

As an example, a security agent (decision-maker, defender) may be screening incoming e-mails. She does not know θ, an indicator of potential security issues associated with them. Her observations could be based on the length of the e-mails, the types of attachments, the presence of certain words, the sending address, etc. A cybercriminal (data-fiddler adversary, attacker) might distort that information through subterfuge, e.g. adding or deleting certain words, using an apparently legal source address and so on, to make her believe that the e-mails are legit and, as a consequence, obtain some benefit.

In this framework, the adversary decides upon his action first, then the decision-maker chooses hers after observing the manipulated data and, finally, both agents take in their losses. In general, the adversary is presumed to allow for some loss related to the resources or effort spent in disturbing the information. This is reflected by the dependence of his loss on his implemented action. In the e-mail screening example, the attacker incurs in monetary costs to purchase an IP, spends some time to appropriately craft the e-mail and commits a crime when forging it, all constituting real or potential losses. Opportunity costs may all also be taken into account through this dependence.

The problem that the decision-maker needs to solve is described in the ID in Figure 4.1a. Not knowing her adversary's choice, as customary, decision node (A) turns out as random to her. In a standard decision-theoretic approach, the decision-maker would solve

$$d^*(y) = \arg\min_d \int l_D(d,\theta)\, p_D(\theta \,|\, y)\, \mathrm{d}\theta. \tag{4.1}$$

Bayes' formula may be used to obtain

$$p_D(\theta \,|\, y) = \frac{p_D(\theta, y)}{p_D(y)} = \frac{1}{p_D(y)} \iint p_D(y \,|\, x, a)\, p_D(x \,|\, \theta)\, p_D(\theta)\, p_D(a)\, \mathrm{d}x\, \mathrm{d}a, \tag{4.2}$$

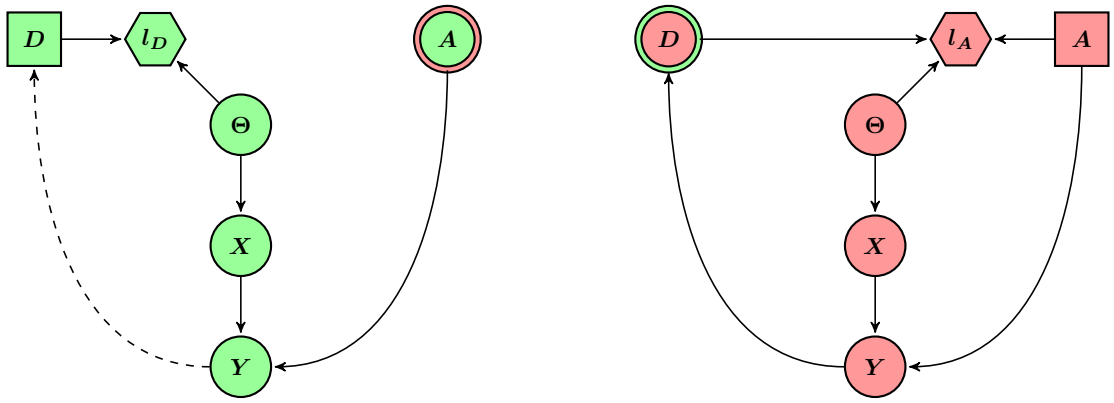
so her optimal decision is equivalent to

$$d^*(y) = \arg\min_d \iiint l_D(d,\theta)\, p_D(y \,|\, x, a)\, p_D(x \,|\, \theta)\, p_D(\theta)\, p_D(a)\, \mathrm{d}x\, \mathrm{d}\theta\, \mathrm{d}a.$$

Of all the assessments required to evaluate the ID, $l_D(d,\theta)$, $p_D(y \,|\, x, a)$, $p_D(x \,|\, \theta)$ and $p_D(\theta)$ are standard in Bayesian SDT. The only distinctive one is $p_D(a)$ (the decision-maker's forecast over the action a) which entails strategic thinking. The ARA approach to ASDT determines it by focusing on the problem that the adversary solves, represented in the ID in Figure 4.1b. This analysis assumes that he wants to minimise his expected loss. For his decision-theoretic solution, he solves

$$a^* = \arg\min_a \iiiint l_A(d,\theta,a)\, p_A(d \,|\, y)\, p_A(y \,|\, x, a)\, p_A(x \,|\, \theta)\, p_A(\theta)\, \mathrm{d}d\, \mathrm{d}y\, \mathrm{d}x\, \mathrm{d}\theta. \tag{4.3}$$

Note that, unlike the decision-maker's, her adversary's actions are taken before making any observations.



(a) Decision-maker's problem

(b) Adversary's problem

Figure 4.1: Both agents' IDs for a data-fiddler adversary in ASDT

As usual, the decision-maker lacks knowledge about the loss function and probabilities used by her adversary. Modelling her uncertainty about them through random losses and probabilities $F \sim \big(L_A(d,\theta,a), P_A(d\,|\,y), P_A(y\,|\,x,a), P_A(x\,|\,\theta), P_A(\theta)\big)$, she would solve

$$A^* = \arg\min_a \iiiint L_A(d,\theta,a)\, P_A(d\,|\,y)\, P_A(y\,|\,x,a)\, P_A(x\,|\,\theta)\, P_A(\theta)\,\mathrm{d}d\,\mathrm{d}y\,\mathrm{d}x\,\mathrm{d}\theta$$

to find the optimal random decision A^*. Thus, the decision-maker comes up with the distribution $p_D(a) = \Pr(A^* = a)$ in the discrete case and, similarly, in the continuous one, that she needs to ascertain her best decision $d^*(y)$. All of her uncertainty about her adversary's situation is properly embodied in it.

K samples $\big(L_A^k(d,\theta,a), P_A^k(d\,|\,y), P_A^k(y\,|\,x,a), P_A^k(x\,|\,\theta), P_A^k(\theta)\big)$, $k = 1,\ldots,K$, would be drawn from F to approximate $p_D(a)$ through simulation, in line with Section 3.3.3, procuring

$$A_k^* = \arg\min_a \iiiint L_A^k(d,\theta,a)\, P_A^k(d\,|\,y)\, P_A^k(y\,|\,x,a)\, P_A^k(x\,|\,\theta)\, P_A^k(\theta)\,\mathrm{d}d\,\mathrm{d}y\,\mathrm{d}x\,\mathrm{d}\theta$$

and estimating

$$\hat{p}_D(a) \approx \#\{A_k^* = a\}/K.$$

As in Chapters 2 and 3, four of the components of F are relatively easy to model, see Banks et al. [2015]:

- $P_A(\theta)$ could be based on $p_D(\theta)$ incorporating some uncertainty about it. For example, should $p_D(\theta)$ be a discrete distribution, $P_A(\theta)$ could be modelled as a Dirichlet distribution with mean $p_D(\theta)$, employing the same techniques presented in Section 2.4.2. Similarly, should $p_D(\theta)$ be a continuous distribution, $P_A(\theta)$ could be modelled as a Dirichlet process with base measure $p_D(\theta)$.

- Analogously, $P_A(y \mid x, a)$ could be built from $p_D(y \mid x, a)$ with some uncertainty.
- This would also apply to $P_A(x \mid \theta)$, which could be formed on $p_D(x \mid \theta)$ with additional uncertainty about it (although in many cases it will be reasonable to accept that they genuinely coincide).
- For $L_A(d, \theta, a)$, one could normally reflect upon the adversary's interests, formulate a parametric form for the loss function and appraise a subjective distribution over its parameters, as conducted in Section 2.4.2.

On the other hand, it is not easy to assess $P_A(d \mid y)$, since it demands strategic thinking and the decision-maker is required to conceive her adversary's convictions on what decision she will support given that she observes y. This predicament was discussed in Section 3.3.3.

4.2 The elementary AHT problem

The ASDT framework in the preceding section is now adapted to the hypothesis testing context. Focus shall be on the problem of testing two simple hypotheses described by $\Theta = \{\theta_0, \theta_1\}$. Continuing with the e-mail screening example, suppose the defender needs to decide whether a batch of e-mails includes spam or not. She has beliefs about the standard stream of legit and spam messages, but an attacker perturbs such stream by adding, deleting or modifying some of the messages, in an attempt to confound her and obtain some benefit. Both agents get different rewards depending on whether the batch is accepted by the defender and includes any spam.

The backbone structure of the AHT problem coincides with that of a data-fiddler adversary scenario in the ASDT problem, covered in Section 4.1. Depending on the uncertain hypothesis $\theta \in \Theta$, there will be an observation data-stream x which gets perturbed to y by the adversary's action a. The modified data-stream y is the one perceived by the decision-maker, who needs to decide which is the relevant hypothesis θ. She makes such decision d without observing neither x nor θ. Depending on d and the actual hypothesis θ, both agents receive their corresponding losses. Besides, the adversary is assumed to spend some effort in performing the attack, as reflected by the dependence of his loss on the attack he implements. The aim is to support the decision-maker in choosing the appropriate hypothesis.

4.2.1 Solving the decision-maker's problem

The problem the decision-maker needs to solve was depicted in Figure 4.1a. Her decision space is $\{d_0, d_1\}$, with d_j denoting her support for θ_j, $j = 0, 1$. Following

a standard Bayesian decision-theoretic approach, suppose that the decision-maker may elicit the following judgements:

- At node Θ, $p_D(\theta)$ models her beliefs about the various hypotheses, which are designated
$$p_D(\theta = \theta_j) = \pi_D^j, \qquad j = 0, 1,$$
with $\pi_D^j \geq 0$ and $\pi_D^0 + \pi_D^1 = 1$.
- At node X, $p_D(x \,|\, \theta)$ expresses her opinion about how data x would depend on each hypothesis θ_j, $j = 0, 1$.
- At node Y, $p_D(y \,|\, x, a)$ describes her understanding about how data will be perturbed: it reflects her notion about what would the observed y be, if x is the relevant data and a is her adversary's selected action.
- At node A, $p_D(a)$ portrays her convictions about which attack a would be undertaken by her adversary.
- At node l_D, $l_D(d, \theta)$ represents the decision-maker's loss function. A standard 0-1-c_D loss function as in Table 4.1 is used, where 0 is the best loss (associated with a system functioning as expected) and 1 is the worst (related to a non-functioning system). Thus, it is assumed that $0 \leq c_D \leq 1$.

		Actual Hypothesis	
		θ_0	θ_1
D's Decision	d_0	0	1
	d_1	c_D	0

Table 4.1: Decision-maker's loss function

The decision-maker would then solve (4.1) getting

$$\arg\min_d \sum_{j=0}^{1} l_D(d, \theta_j)\, p_D(\theta_j \,|\, y).$$

After simple computations, it follows that her optimal decision would be to support θ_0 if and only if

$$p_D(\theta_1 \,|\, y) \leq c_D\, p_D(\theta_0 \,|\, y).$$

Making use of (4.2) results in

$$p_D(\theta_j \,|\, y) = \frac{\pi_D^j}{p_D(y)} \iint p_D(y \,|\, x, a)\, p_D(x \,|\, \theta_j)\, p_D(a)\, \mathrm{d}x\, \mathrm{d}a, \qquad j = 0, 1.$$

Therefore, the optimal decision for the decision-maker would be d_0 if and only if

$$\begin{aligned} \pi_D^1 \iint p_D(y \,|\, x, a)\, p_D(x \,|\, \theta_1)\, p_D(a)\, \mathrm{d}x\, \mathrm{d}a \\ \leq \\ c_D\, \pi_D^0 \iint p_D(y \,|\, x, a)\, p_D(x \,|\, \theta_0)\, p_D(a)\, \mathrm{d}x\, \mathrm{d}a. \end{aligned} \tag{4.4}$$

As in Section 4.1, among the required assessments, the one related to $p_D(a)$ is non-standard and involves strategic thinking. Its estimation is facilitated by considering the problem that her adversary should elucidate.

4.2.2 Modelling the adversary's problem

Figure 4.1b provided the adversary's decision-making problem, assuming that he aims at minimising expected loss. For its decision-theoretic solution, the adversary would need the subsequent appreciations:

- At node Θ, $p_A(\theta)$ models his beliefs about the likelihood of the hypotheses, which are designated
$$p_A(\theta = \theta_j) = \pi_A^j, \qquad j = 0, 1,$$
with $\pi_A^j \geq 0$ and $\pi_A^0 + \pi_A^1 = 1$.
- At node X, $p_A(x \,|\, \theta)$ expresses his opinion about the data-stream x, for each hypothesis θ_j, $j = 0, 1$.
- At node Y, $p_A(y \,|\, x, a)$ describes his understanding about what would the effect of his actions a be in transforming the data x to y.
- At node D, $p_A(d \,|\, y)$ portrays his convictions about the decision-maker's decision d provided that she observes y.
- At node l_A, $l_A(d, \theta, a)$ represents the adversary's loss function, structured as in Table 4.2. Typically, $l_{00}(a) \geq l_{01}(a)$ and $l_{10}(a) \leq l_{11}(a)$, since it is better for him when the defender makes mistakes.

		Actual Hypothesis	
		θ_0	θ_1
D's Decision	d_0	$l_{00}(a)$	$l_{01}(a)$
	d_1	$l_{10}(a)$	$l_{11}(a)$

Table 4.2: Adversary's loss function, given attack a

An important case is described in Table 4.3, where $0 \leq c_A^0 \leq c_A^1 \leq 1$ is assumed. This reflects that 0 is the adversary's best loss (attained when the decision-maker supports θ_0 and she should not), while 1 is his worst (obtained when the decision-maker upholds θ_0 and she should), The intermediate cases manifest that it is worse for the adversary that the decision-maker favours θ_1 when the actual hypothesis is θ_1 (incurring in costs and motivating security awareness) than when it is θ_0 (avoiding costs and inducing disproportionate security exigence).

		Actual Hypothesis	
		θ_0	θ_1
D's Decision	d_0	1	0
	d_1	c_A^0	c_A^1

Table 4.3: Adversary's loss function

Should the above assessments be available, plugging them in (4.3), the optimal decision a^* for the adversary would be

$$a^* = \arg\min_a \sum_{i,j=0}^{1} \pi_A^j \iint l_A(d_i, \theta_j, a)\, p_A(d_i \,|\, y)\, p_A(y \,|\, x, a)\, p_A(x \,|\, \theta_j)\, \mathrm{d}y\, \mathrm{d}x.$$

However, as common knowledge is not applicable, the decision-maker resorts to random losses and probabilities as in Section 4.1 and solves instead

$$A^* = \arg\min_a \sum_{i,j=0}^{1} \Pi_A^j \iint L_A(d_i, \theta_j, a)\, P_A(d_i \,|\, y)\, P_A(y \,|\, x, a)\, P_A(x \,|\, \theta_j)\, \mathrm{d}y\, \mathrm{d}x. \quad (4.5)$$

4.2.3 AHT: A numerical example

The previous ideas for the AHT problem are illustrated with a numerical example in which a defender (decision-maker) monitors continuous positive observations perturbed by an attacker (adversary). The two entertained hypotheses are $\theta_0 = 2$ and $\theta_1 = 1$. Required elements are displayed as introduced in Section 4.2.1 for the defender's problem:

- As for the priors over the hypotheses, both are assumed to be equally likely a priori, so that $\pi_D^0 = \pi_D^1 = \frac{1}{2}$.

- With regard to $p_D(x \mid \theta)$, the defender receives data x exponentially distributed $\mathcal{E}xp(\theta_j)$, with uncertainty about the parameter θ_j, $j = 0, 1$.
- The attacker may modify data x according to a strategy which allows for keeping, doubling or halving its value. Such actions shall be respectively denoted a_0, a_1 and a_2: if x is the actual value, the defender will perceive $y = x$ if the attacker chooses a_0, whereas $y = 2x$ and $y = x/2$ will be the observed values if the attacker chooses a_1 and a_2, respectively. Therefore, distributions $p_D(y \mid x, a)$ are Dirac measures with support at $y = x$ (a_0), $y = 2x$ (a_1) and $y = x/2$ (a_2).
- As an illustration, start by contemplating the case in which the defender knows the probabilities $p_D(a)$ with which the attacker chooses among his actions. Suppose, for the moment, that $p_D(a_0) = \frac{1}{2}$, $p_D(a_1) = \frac{1}{6}$ and $p_D(a_2) = \frac{1}{3}$.
- The loss function in Table 4.1 is considered for $l_D(d, \theta)$, with $c_D = \frac{3}{4}$.

Recall (4.4), leading the defender to adopt decision d_0 (accept θ_0). In this case, using $l_D(d, \theta)$, $p_D(x \mid \theta)$ and $p_D(y \mid x, a)$, such condition becomes

$$\pi_D^1 \left[\theta_1\, e^{-\theta_1 y}\, p_D(a_0) + \theta_1\, e^{-\theta_1 \frac{y}{2}}\, p_D(a_1) + \theta_1\, e^{-\theta_1 2y}\, p_D(a_2)\right] \leq \frac{3}{4}\, \pi_D^0 \left[\theta_0\, e^{-\theta_0 y}\, p_D(a_0) + \theta_0\, e^{-\theta_0 \frac{y}{2}}\, p_D(a_1) + \theta_0\, e^{-\theta_0 2y}\, p_D(a_2)\right].$$

Substituting the specific values of θ_0, θ_1, π_D^0 and π_D^1, results in

$$\frac{1}{2} \left[p_D(a_0)\, e^{-y} + p_D(a_1)\, e^{-\frac{y}{2}} + p_D(a_2)\, e^{-2y}\right] \leq \frac{3}{8} \left[2p_D(a_0)\, e^{-2y} + 2p_D(a_1)\, e^{-y} + 2p_D(a_2)\, e^{-4y}\right]. \tag{4.6}$$

Finally, the incorporation of the particular $p_D(a)$ produces

$$\frac{1}{2} \left[\frac{1}{2} e^{-y} + \frac{1}{6} e^{-\frac{y}{2}} + \frac{1}{3} e^{-2y}\right] \leq \frac{3}{8} \left[e^{-2y} + \frac{1}{3} e^{-y} + \frac{2}{3} e^{-4y}\right],$$

which gets simplified to checking the inequality

$$2e^{-\frac{y}{2}} + 3e^{-y} - 5e^{-2y} - 6e^{-4y} \leq 0.$$

It may be estimated that decision d_0 should be made when a value $y \lesssim 0.372$ is observed. However, a slight change of the parameters might deliver a completely different result. For example, with $\pi_D^0 = \frac{1}{3}$ (and $\pi_D^1 = \frac{2}{3}$), and all other probabilities and costs as above, d_1 is optimal regardless of the observed y. In addition, note that an adversary-unaware defender would expect $p_D(a_0) = 1$ (and $p_D(a_1) = p_D(a_2) = 0$),

alternatively leading in (4.6) to support d_0 when a value $y \lesssim 0.405$ is perceived and, thus, biasing the choice in favour of θ_0.

Consider now the case in which the defender does not accurately know $p_D(a)$. To comply with the ARA approach, assume that the following assessments from Section 4.2.2 are made:

- The defender considers that $\Pi_A^0 \sim \mathcal{U}(0.25, 0.75)$ is drawn uniformly (and $\Pi_A^1 = 1 - \Pi_A^0$).

- The defender's knowledge of $P_A(x \,|\, \theta)$, where $\theta \in \{\theta_0, \theta_1\}$, is modelled as a Gamma distribution $\mathcal{G}a(\alpha, \beta)$ with mean $\theta = \alpha/\beta$ and variance $\sigma^2 = \alpha/\beta^2 \sim \mathcal{U}(0.5, 2)$ uniformly distributed. This variance randomness induces that of $P_A(x \,|\, \theta)$.

- The $P_A(y \,|\, x, a)$ distributions will be Dirac, coinciding with the defender's $p_D(y \,|\, x, a)$.

- Distribution $P_A(d \,|\, y)$ is built based on the likelihood $h(y \,|\, d, a)$ of y under different choices of d and a, mixing them through a random allocation of probabilities to each action. Suppose that the defender assumes that the attacker thinks she is modelling the data with an exponential distribution, assessing the probabilities $(\epsilon_0, \epsilon_1, \epsilon_2)$ assigned by him to each strategy through a Dirichlet distribution $\mathcal{D}ir(1, 1, 1)$. Then, $P_A(d_0 \,|\, y)$ has the subsequent form:

$$g(\epsilon_0, \epsilon_1, \epsilon_2, y) = \frac{\sum_{j=0}^{2} \epsilon_j \, h(y \,|\, d_0, a_j)}{\sum_{j=0}^{2} \epsilon_j \, h(y \,|\, d_0, a_j) + \sum_{j=0}^{2} \epsilon_j \, h(y \,|\, d_1, a_j)}$$

$$= \frac{2\left(\epsilon_0 \, e^{-2y} + \epsilon_1 \, e^{-y} + \epsilon_2 \, e^{-4y}\right)}{2\left(\epsilon_0 \, e^{-2y} + \epsilon_1 \, e^{-y} + \epsilon_2 \, e^{-4y}\right) + \epsilon_0 \, e^{-y} + \epsilon_1 \, e^{-\frac{y}{2}} + \epsilon_2 \, e^{-2y}}.$$

  The distribution of $(\epsilon_0, \epsilon_1, \epsilon_2)$ causes the randomness of $P_A(d_0 \,|\, y)$. Finally, $P_A(d_1 \,|\, y) = 1 - P_A(d_0 \,|\, y)$.

- The random loss function $L_A(d, \theta, a)$ is as in Table 4.3, where $C_A^0 \equiv 0$ is degenerate and $C_A^1 \sim \mathcal{U}(0.5, 1)$ is uniformly distributed.

Taking into account the characterisation of $L_A(d, \theta, a)$ and $P_A(y \,|\, x, a)$ in (4.5), the attacker's random expected losses for his three actions will be:

$$\Psi_A(a_0) = \Pi_A^0 \int P_A(d_0 \,|\, y = x)\, P_A(x \,|\, \theta_0)\, \mathrm{d}x + C_A^1\, \Pi_A^1 \int P_A(d_1 \,|\, y = x)\, P_A(x \,|\, \theta_1)\, \mathrm{d}x,$$

$$\Psi_A(a_1) = \Pi_A^0 \int P_A(d_0 \,|\, y = 2x)\, P_A(x \,|\, \theta_0)\, \mathrm{d}x + C_A^1\, \Pi_A^1 \int P_A(d_1 \,|\, y = 2x)\, P_A(x \,|\, \theta_1)\, \mathrm{d}x,$$

$$\Psi_A(a_2) = \Pi_A^0 \int P_A(d_0 \,|\, y = \tfrac{x}{2})\, P_A(x \,|\, \theta_0)\, \mathrm{d}x + C_A^1\, \Pi_A^1 \int P_A(d_1 \,|\, y = \tfrac{x}{2})\, P_A(x \,|\, \theta_1)\, \mathrm{d}x.$$

The random models for $L_A(d, \theta, a)$, $P_A(\theta)$, $P_A(x \,|\, \theta)$ and $P_A(d \,|\, y)$ induce the randomness in these expected losses. The attack probabilities are estimated as follows:

Algorithm 5 AHT: Simulation of attack probabilities

Data: Hypotheses θ_0 and θ_1; number of iterations K.

Set $p_j = 0, \quad j = 0, 1, 2.$

For $k = 1$ **to** K **do**

Generate $\pi_A^{0,k} \sim \mathcal{U}(0.25, 0.75)$ and compute $\pi_A^{1,k} = 1 - \pi_A^{0,k}$.

Generate $\sigma_{i,k}^2 \sim \mathcal{U}(0.5, 2)$ and compute $\alpha_i^k = \theta_i^2/\sigma_{i,k}^2$; $\beta_i^k = \theta_i/\sigma_{i,k}^2, \quad i = 0, 1.$

Generate $(\epsilon_0^k, \epsilon_1^k, \epsilon_2^k) \sim \mathcal{D}ir(1, 1, 1)$ and $C_A^{1,k} \sim \mathcal{U}(0.5, 1)$.

$$\Psi_A^k(a_0) = \pi_A^{0,k} \int (1 - g(\epsilon_0^k, \epsilon_1^k, \epsilon_2^k, x))\, f(x \,|\, \alpha_0^k, \beta_0^k)\, \mathrm{d}x + C_A^{1,k}\, \pi_A^{1,k} \int g(\epsilon_0^k, \epsilon_1^k, \epsilon_2^k, x)\, f(x \,|\, \alpha_1^k, \beta_1^k)\, \mathrm{d}x.$$

$$\Psi_A^k(a_1) = \pi_A^{0,k} \int (1 - g(\epsilon_0^k, \epsilon_1^k, \epsilon_2^k, 2x))\, f(x \,|\, \alpha_0^k, \beta_0^k)\, \mathrm{d}x + C_A^{1,k}\, \pi_A^{1,k} \int g(\epsilon_0^k, \epsilon_1^k, \epsilon_2^k, 2x)\, f(x \,|\, \alpha_1^k, \beta_1^k)\, \mathrm{d}x.$$

$$\Psi_A^k(a_2) = \pi_A^{0,k} \int (1 - g(\epsilon_0^k, \epsilon_1^k, \epsilon_2^k, x/2))\, f(x \,|\, \alpha_0^k, \beta_0^k)\, \mathrm{d}x + C_A^{1,k}\, \pi_A^{1,k} \int g(\epsilon_0^k, \epsilon_1^k, \epsilon_2^k, x/2)\, f(x \,|\, \alpha_1^k, \beta_1^k)\, \mathrm{d}x.$$

Find $j^* = \arg\min_j \ \Psi_A^k(a_j)$.

Set $p_{j^*} = p_{j^*} + 1$.

End For

Set $\hat{p}_D(a_j) = p_j/K, \quad j = 0, 1, 2.$

An application in `R` (Appendix B.3) of Algorithm 5 with $K = 10^5$ iterations leads to estimates $\hat{p}_D(a_0) \approx 0.04$, $\hat{p}_D(a_1) \approx 0.85$ and $\hat{p}_D(a_2) \approx 0.11$. Plugging such values in (4.6), the optimal decision is d_0 when a value $y \lesssim 0.737$ is observed, which differs from the Bayesian solution obtained earlier.

4.2.4 A game-theoretic perspective

In order to provide a broader understanding of the benefits of the ARA approach to the AHT problem, a standard game-theoretic perspective to it is presented here. The numerical example in the previous section is employed for this purpose.

Within the game-theoretic framework, both agents' loss functions are redefined in terms of their combined decisions. If the defender's decision is d and the attacker's is a, the defender's (equivalent) expected payoff is

$$\psi_D(d, y, a) = \sum_{j=0}^{1} \pi_D^j \int l_D(d, \theta_j)\, p_D(y \,|\, x, a)\, p_D(x \,|\, \theta_j)\, \mathrm{d}x,$$

and that of the attacker is

$$\psi_A(d, a) = \sum_{j=0}^{1} \pi_A^j\, l_A(d, \theta_j, a).$$

Under common knowledge assumptions, if a Nash equilibrium $(d^*(y), a^*)$ exists, then it must satisfy

$$\psi_D(d^*(y), y, a^*) \leq \psi_D(d, y, a^*), \qquad \forall d \in \{d_0, d_1\};$$
$$\psi_A(d^*(y), a^*) \leq \psi_D(d^*(y), a), \qquad \forall a \in \{a_0, a_1, a_2\}.$$

When replicating the numerical example in Section 4.2.3, the elements involved in the defender's problem coincide, while those required for the attacker's problem shall be based on the means of the defender's assessments of the specified probability distributions. Thus, $\pi_A^0 = E\left[\mathcal{U}(0.25, 0.75)\right] = \frac{1}{2} = \pi_A^1$, $c_A^0 = 0$ and $c_A^1 = E\left[\mathcal{U}(0.5, 1)\right] = \frac{3}{4}$. The defender's expected payoff is determined then as

$$\psi_D(d_0, y, a_0) = \tfrac{1}{2}e^{-y}, \qquad \psi_D(d_0, y, a_1) = \tfrac{1}{2}e^{-\frac{y}{2}}, \qquad \psi_D(d_0, y, a_2) = \tfrac{1}{2}e^{-2y},$$
$$\psi_D(d_1, y, a_0) = \tfrac{3}{4}e^{-2y}, \qquad \psi_D(d_1, y, a_1) = \tfrac{3}{4}e^{-y}, \qquad \psi_D(d_1, y, a_2) = \tfrac{3}{4}e^{-4y}.$$

That of the attacker is

$$\psi_A(d_0, a) = \tfrac{1}{2}, \qquad \psi_D(d_1, a) = \tfrac{3}{8}, \qquad \forall a \in \{a_0, a_1, a_2\}.$$

Only a Bayes-Nash equilibrium may be found. The attacker would rely on any of his three possible attacks (a_0, a_1 and a_2) with probability $\frac{1}{3}$; the defender, also with probability $\frac{1}{3}$, would follow one of the three corresponding matching decision rules: choose d_0 if $y \lesssim 0.405$ (R0), choose d_0 if $y \lesssim 0.811$ (R1) and choose d_0 if $y \lesssim 0.203$ (R2).

The defender could deviate from the mixed-strategies Nash equilibrium and just adopt one of the decision rules in terms of her risk attitude towards adopting d_0, where R2 is the most conservative, R1 the least and R0 in-between. In any case, the obtained game-theoretic decision rules do not aggregate all of her uncertainty, but rather condense it around the different attacker's choices. On the contrary, the ARA solution (4.6) provides the defender with a precise decision rule that takes into account both her aleatory and epistemic uncertainty. Moreover, ideas in Chapter 3 could be applied to deal with concept uncertainty and even combine, among others, the Bayes-Nash equilibrium solution with the level-2 thinking ARA solution developed in this chapter.

4.3 A batch acceptance model

As a significant example of how the structure of the elementary AHT problem may be extended to more sophisticated situations, a model for batch acceptance is considered. The problem dealt with is deciding whether to accept a batch of items received over a period of time among which some of them could be faulty, thus entailing potential security and/or performance problems. This type of issues arise in areas such as screening containers at international ports, filtering batches of electronic messages or admitting packages of perishable products or electronic components. The main difference with the AHT model in Section 4.2 is that, in this case, the decision-maker has only partial information on the observations and the consequences of her decision do not depend explicitly on the relevant hypothesis but on the observed data. A non-adversarial version of the problem is first described, which is afterwards modified to include adversaries.

4.3.1 Problem setting

The problem initially faced is outlined in Figure 4.2. Its structure is similar to the SDT problem depicted in Figure 3.1, except for two key differences. One is minor, since two influencing states are considered, called θ and γ, as well as two pieces of information: the batch size m (observed when making the decision) and its composition (unobserved when making the decision) with x acceptable items and $m - x$ faulty items. The second one is major, since the relevant hypothesis (the

batch acceptability) does not directly rely on parameters θ and γ, but is rather determined by data m and x (in particular, by the presence of faulty items).

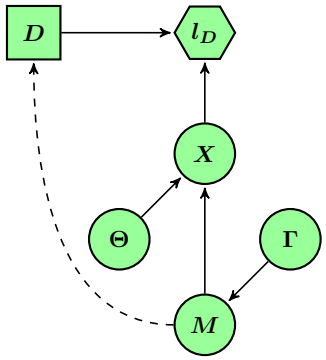


Figure 4.2: Outline of the non-adversarial batch acceptance problem

The problem is specified as follows:

- A defender (decision-maker) receives a batch with two types of items (X): 0, associated with acceptable items; and 1, corresponding to faulty ones. She needs to decide (D) whether to accept (d_0) or reject (d_1) the batch.

- The defender observes the size of the batch (M), which is related with certain parameter (Γ). To fix ideas, assume that, over a period of duration 1, the number of items m follows a Poisson distribution with an average of γ items so that $m \,|\, \gamma \sim \mathcal{P}o(\gamma)$. In addition, suppose that the prior over γ is a Gamma distribution $\mathcal{G}a(a, b)$. After t periods in which, in total, r items have arrived, the posterior is $\gamma \,|\, t, r \sim \mathcal{G}a(a + r, b + t)$. Note that γ will have no impact in the non-adversarial version, as the defender observes the actual value of m. However, it will provide useful information about m in the adversarial case in Section 4.3.2.

- The probability that an item is acceptable is determined by another parameter (Θ). If Z is used to designate this ($z = 0$, an acceptable item; $z = 1$, otherwise), $p_D(z = 0 \,|\, \theta) = \theta$. The number of acceptable items x among the total amount m will have a binomial distribution $x \,|\, m, \theta \sim \mathcal{B}in(m, \theta)$. To complete model specification, consider that prior beliefs about θ are modelled through a Beta distribution $\mathcal{B}e(\alpha, \beta)$. If after receiving r items, s have been acceptable (and $r - s$, faulty), the posterior is $\theta \,|\, r, s \sim \mathcal{B}e(\alpha + s, \beta + r - s)$.

As for the loss function l_D, numerous forms may be contemplated. Two are mentioned here, although only the first one shall be used in the adversarial problem in Section 4.3.2.

Form A: Winner takes it all

A batch with m items is received in a given period. In this scenario, just allowing one faulty item is as bad as allowing several of them, because of the entailed security and/or performance issues. The loss composition is displayed in Table 4.4, where c describes the (expected) opportunity costs associated with rejecting a batch with all acceptable items.

		Batch of m Items		
		All Acceptable	**Some Faulty**	
		$\boldsymbol{\theta^m}$	$\mathbf{1-\theta^m}$	**Exp. Loss**
D's Decision	**Accept, d_0**	0	1	$1-\theta^m$
	Reject, d_1	c	0	$c\,\theta^m$

Table 4.4: Defender's loss function (A)

The expected losses of decisions d_0 (accept) and d_1 (reject) are

$$\psi_D(d_0) = E_\theta\left[1-\theta^m\right] = 1 - E_\theta\left[\theta^m\right], \qquad \psi_D(d_1) = E_\theta\left[c\,\theta^m\right] = c\,E_\theta\left[\theta^m\right].$$

Then, the defender's optimal decision is to accept the batch (d_0) if and only if

$$1 - E_\theta\left[\theta^m\right] \leq c\,E_\theta\left[\theta^m\right] \quad \Longleftrightarrow \quad E_\theta\left[\theta^m\right] \geq \frac{1}{1+c}.$$

Since $E_\theta\left[\theta^m\right]$ decreases as m increases, there will be a threshold value m_A such that if $m > m_A$, the decision would be to reject the batch (d_1). In particular, with the posterior $\mathcal{B}e(\alpha+s, \beta+r-s)$ model for θ, it follows that

$$E_\theta\left[\theta^m\right] = \prod_{k=0}^{m-1} \frac{\alpha+s+k}{\alpha+\beta+r+k}, \tag{4.7}$$

and m_A may be recursively obtained.

Form B: Each fault counts

In this case, the loss will depend on the number $m-x$ of faulty items included in the batch, because of the increased security and/or performance problems. The relevant loss structure is displayed in Table 4.5, where the new parameter c' reflects the (expected) loss per faulty item accepted.

		Batch of m Items		
		All Acceptable	**x Acceptable**	
		$\boldsymbol{\theta^m}$	$\binom{m}{x}\,\boldsymbol{\theta^x\,(1-\theta)^{m-x}}$	**Exp. Loss**
D's Decision	**Accept, d_0**	0	$(m-x)\,c'$	$m\,c'\,(1-\theta)$
	Reject, d_1	c	0	$c\,\theta^m$

Table 4.5: Defender's loss function (B)

The expected losses of both decisions are

$$\psi_D(d_0) = E_\theta\left[m\,c'\,(1-\theta)\right] = m\,c'\,(1-E_\theta\left[\theta\right]), \qquad \psi_D(d_1) = E_\theta\left[c\,\theta^m\right] = c\,E_\theta\left[\theta^m\right].$$

Therefore, the defender's optimal action is to accept the batch (d_0) if and only if

$$m\,c'\,(1-E_\theta\left[\theta\right]) \leq c\,E_\theta\left[\theta^m\right] \quad \Longleftrightarrow \quad \frac{E_\theta\left[\theta^m\right]}{m} \geq \frac{c'}{c}\,(1-E_\theta\left[\theta\right]).$$

As with the previous loss function's form, since $E_\theta\left[\theta^m\right]$ decreases as m increases, there will be a threshold value m_B such that if $m > m_B$, the decision would be to reject the batch (d_1). Specifically, with the posterior $\mathcal{B}e(\alpha+s, \beta+r-s)$ model for θ, the decision is to accept the batch (d_0) if and only if

$$\frac{E_\theta\left[\theta^m\right]}{m} \geq \frac{c'}{c}\,\frac{\beta+r-s}{\alpha+\beta+r},$$

where, once again, (4.7) may be used to find m_B recursively.

4.3.2 Adversarial problem

The adversarial version is now engaged, considering the loss in Section 4.3.1 (Form A). As represented in the BAID in Figure 4.3, an attacker (adversary) is faced who might alter the received batch so as to confound the defender to reach some objectives.

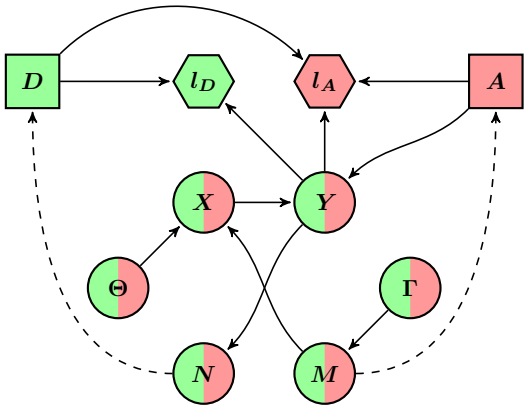
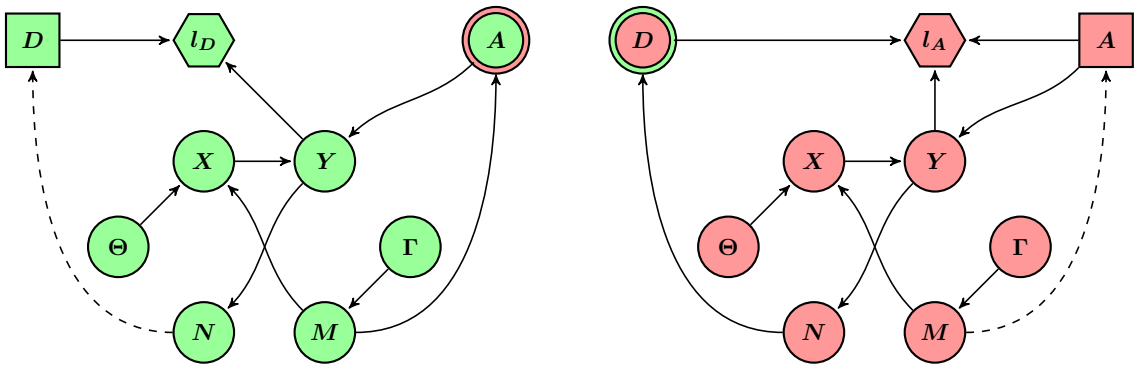


Figure 4.3: ARA modelling of the batch acceptance problem

The original batch x is influenced by parameters γ, which regulates the number m of items received, and θ, which conditions their quality. The attacker knows the size m of the batch before choosing his attack, possibly modifying the size of the final batch y to n, which is observed by the defender before making her decision. The problems of the defender and attacker are, respectively, displayed in Figures 4.4a and 4.4b.



(a) Defender's problem

(b) Attacker's problem

Figure 4.4: Both agents' IDs for the batch acceptance problem

Three possible attack Tactics T1, T2 and T3 are gradually studied, identifying the attacker's decision variables, how the item arrival process changes, the attacker's loss function and the solution. The final number of items in a batch will be n, with x acceptable items and $m-x$ original faulty ones, designated outer faults (O-faults). The remaining $n-m$ items correspond to faulty items introduced by the attacker, called A-faults. The attacker's loss is smaller (greater benefit) if the defender accepts an A-fault rather than an O-fault.

T1: A-fault injection

Within this tactic, the attacker injects y_1 of his faulty items. The data received by the defender includes x acceptable items, $m - x$ O-faults and y_1 A-faults. The attacker needs to decide y_1, which is random to the defender. As announced in Section 4.3.1, γ will be relevant here, since it provides information about m.

Suppose first that the defender knows the distribution $p_D(y_1 \,|\, m)$, describing her beliefs about how many faulty items will be injected by the attacker if the original batch size were m. The loss function for the defender is as in Table 4.6, where the probability of having a final batch size of $n = m + y_1$ items, given γ, is

$$p_1(n \,|\, \gamma) = \sum_{i=0}^{n} p_D(y_1 = n - i \,|\, m = i)\, p_D(m = i \,|\, \gamma),$$

reflecting the possible initial sizes of the batch and the included faulty items. The probability that all those items are acceptable ($x = m$ and $y_1 = 0$) is

$$q_1(n \,|\, \gamma) = \frac{p_D(y_1 = 0 \,|\, m = n)\, p_D(m = n \,|\, \gamma)}{p_1(n \,|\, \gamma)}\, \theta^n,$$

since the only combination for an acceptable final batch is having n initial acceptable items ($x = m = n$) and no faulty items injected ($y_1 = 0$).

		Final Batch of n Items		
		All Acceptable	**Some Faulty**	
		$\boldsymbol{q_1(n \,\vert\, \gamma)}$	$\boldsymbol{1 - q_1(n \,\vert\, \gamma)}$	**Exp. Loss**
D's Decision	**Accept, d_0**	0	1	$1 - q_1(n \,\vert\, \gamma)$
	Reject, d_1	c	0	$c\, q_1(n \,\vert\, \gamma)$

Table 4.6: Defender's loss function (T1)

The expected losses of decisions d_0 (accept) and d_1 (reject) are

$$\psi_D(d_0) = 1 - E_{\theta,\gamma}\left[q_1(n \,|\, \gamma)\right], \qquad \psi_D(d_1) = c\, E_{\theta,\gamma}\left[q_1(n \,|\, \gamma)\right].$$

Then, the rule is to accept the batch (d_0) if and only if

$$E_{\theta,\gamma}\left[q_1(n \,|\, \gamma)\right] \geq \frac{1}{1 + c},$$

whose evaluation would typically require simulation.

An ARA procedure is now provided to estimate the crucial quantities $p_D(y_1 \,|\, m)$ and, thus, $q_1(n \,|\, \gamma)$. Consider the attacker's loss function depicted in Table 4.7, which depends on the batch composition and the decision made by the defender, as well as on his own choice. It holds that $x \in \{0, 1, \ldots, m\}$ and $y_1 \in \{0, 1, \ldots\}$. The involved parameters are the expected gain h due to each O-fault, the expected gain g due to each A-fault and the unitary cost f of injecting A-faults.

		Final Batch Composition		
		Acceptable	**O-Fault**	**A-Fault**
		$\boldsymbol{x}$	$\boldsymbol{m - x}$	$\boldsymbol{y_1}$
D's Decision	**Accept, d_0**	0	$-h$	$f - g$
	Reject, d_1	0	0	f

Table 4.7: Attacker's loss per item (T1)

Given that the attacker chooses y_1, his losses associated with both defender's decisions are

$$l_A(d_0, y_1) = -h\,(m - x) + (f - g)\,y_1, \qquad l_A(d_1, y_1) = f\,y_1.$$

Knowing the original batch size m, the attacker selects y_1 to minimise his expected loss, which is

$$\begin{aligned}\Psi_A(y_1 \,|\, m) &= p_A(d_0 \,|\, n = m + y_1) \int \left(\sum_{x=0}^{m} l_A(d_0, y_1)\, p_A(x \,|\, m, \theta) \right) p_A(\theta)\, \mathrm{d}\theta \\ &\quad + (1 - p_A(d_0 \,|\, n = m + y_1))\, l_A(d_1, y_1) \\ &= y_1\,(f - g\, p_A(d_0 \,|\, n = m + y_1)) \\ &\quad - h\, p_A(d_0 \,|\, n = m + y_1) \int \left(\sum_{x=0}^{m} (m - x)\, p_A(x \,|\, m, \theta) \right) p_A(\theta)\, \mathrm{d}\theta,\end{aligned}$$

where $p_A(d_0 \,|\, n = m + y_1)$ reflects his beliefs about the defender's decision being to accept the batch (d_0), given that she perceives the batch size to be $n = m + y_1$.

Since the defender lacks information about the attacker's loss function and probabilities, as usual, she may model her uncertainty about them through random losses and probabilities $\big(F, G, H, P_A(d_0 \,|\, n), P_A(x \,|\, m, \theta), P_A(\theta)\big)$ and look for the random

optimal attack $Y_1^*(m)$ that minimises in y_1

$$y_1 \left(F - G\, P_A(d_0 \,|\, n = m + y_1)\right)$$

$$- H\, P_A(d_0 \,|\, n = m + y_1) \int \left(\sum_{x=0}^{m} (m - x)\, P_A(x \,|\, m, \theta) \right) P_A(\theta)\, \mathrm{d}\theta.$$

Then, she would estimate

$$\hat{p}_D(y_1 \,|\, m) \approx \#\{Y_{1,k}^*(m) = y_1\}/K,$$

where $\{Y_{1,k}^*(m)\}_{k=1}^K$ would be a sample of size K from $Y_1^*(m)$, obtained by drawing from the involved components and computing the corresponding optimal amount of injected faulty items.

Regarding the attacker's random losses and probabilities, typical assumptions would be:

- The gains and costs could be uniformly distributed: $F \sim \mathcal{U}(f_1, f_2)$, $G \sim \mathcal{U}(g_1, g_2)$ and $H \sim \mathcal{U}(h_1, h_2)$.
- $P_A(d_0 \,|\, n)$ could be modelled through a uniform distribution, although this might require further recursions, if deeper strategic thinking is considered, as discussed in Section 3.3.3.
- Due to its specificity, $P_A(x \,|\, m, \theta)$ could actually be regarded as a Binomial distribution $\mathcal{B}in(m, \theta)$, i.e. not a random distribution.
- $P_A(\theta)$ could be a Dirichlet process with a Beta distribution base $\mathcal{B}e(\alpha + s, \beta + r - s)$ and concentration parameter ρ.

T2: Item modification

This tactic corresponds to the attacker modifying y_2 of the original items into faults of his. The data perceived by the defender includes $x - y_2^0$ acceptable items, $m - x - y_2^1$ O-faults and y_2 A-faults, where y_2^0 and y_2^1 verify $y_2^0 + y_2^1 = y_2$ and $0 \leq y_2^0 \leq x$, $0 \leq y_2^1 \leq m - x$. The attacker is supposed to not distinguish the type of items he changes and needs to determine y_2, which is random to the defender.

Pretend first that the defender is acquainted with the distribution $p_D(y_2 \,|\, m)$, describing her beliefs about how many items will be modified by the attacker if the original batch size were m. The defender's loss structure is as in Table 4.6, replacing $q_1(n \,|\, \gamma)$ by $q_2(n)$, defined as follows. To start with, the probability of having a final batch with $n = m$ items, given γ, is

$$p_2(n \,|\, \gamma) = p_D(m = n \,|\, \gamma),$$

indicating the only possible initial size of the batch and the included faulty items. Then, the probability that all those items are acceptable ($x = m$ and $y_2 = 0$) is

$$q_2(n) = p_D(y_2 = 0 \,|\, m = n)\, \theta^n,$$

manifesting that the only combination for an acceptable final batch is having n initial acceptable items ($x = m = n$) and no faulty items included ($y_2 = 0$). In this case, knowing γ is irrelevant for gaining information about the batch configuration, since the initial and final batch sizes coincide.

The expected losses of both decisions are

$$\psi_D(d_0) = 1 - E_\theta\left[q_2(n)\right], \qquad \psi_D(d_1) = c\, E_\theta\left[q_2(n)\right];$$

which may be simplified to

$$\psi_D(d_0) = 1 - p_D(y_2 = 0 \,|\, m = n)\, E_\theta\left[\theta^n\right], \qquad \psi_D(d_1) = c\, p_D(y_2 = 0 \,|\, m = n)\, E_\theta\left[\theta^n\right].$$

Therefore, the defender's optimal choice is to accept the batch (d_0) if and only if

$$p_D(y_2 = 0 \,|\, m = n)\, E_\theta\left[\theta^n\right] \geq \frac{1}{1+c}.$$

A way to estimate $p_D(y_2 \,|\, m)$ and, thus, the crucial quantity $p_D(y_2 = 0 \,|\, m = n)$ under an ARA approach is now indicated. The attacker's loss function is described in Table 4.8, dictated by the batch composition and the decision made by the defender (and the attacker's decision). It holds that $x \in \{0, 1, \ldots, m\}$ and $y_2 = y_2^0 + y_2^1 \in \{0, 1, \ldots, m\}$. The new parameter f' is the cost of changing one item to make it faulty.

		Final Batch Composition		
		Acceptable	**O-Fault**	**A-Fault**
		$\boldsymbol{x - y_2^0}$	$\boldsymbol{m - x - y_2^1}$	$\boldsymbol{y_2}$
D's Decision	**Accept, $\boldsymbol{d_0}$**	0	$-h$	$f' - g$
	Reject, $\boldsymbol{d_1}$	0	0	f'

Table 4.8: Attacker's loss per item (T2)

The attacker's (expected) losses associated with both defender's decisions, when he chooses y_2, are

$$l_A(d_0, y_2) = -h\left(m - x - E\left[y_2^1\right]\right) + (f' - g)\, y_2, \qquad l_A(d_1, y_2) = f'\, y_2.$$

Assuming that he picks the items randomly, so that $E\,[y_2^1] = y_2\,\frac{m-x}{m}$, then

$$l_A(d_0, y_2) = -h\,(m-x)\,(1-\tfrac{y_2}{m}) + (f'-g)\,y_2.$$

The attacker needs to select y_2 so as to minimise his expected loss when the original batch size is m, which is

$$\begin{aligned}\Psi_A(y_2\,|\,m) &= p_A(d_0\,|\,n=m)\int\left(\sum_{x=0}^{m} l_A(d_0,y_2)\,p_A(x\,|\,m,\theta)\right)p_A(\theta)\,\mathrm{d}\theta \\ &\qquad + (1-p_A(d_0\,|\,n=m))\,l_A(d_1,y_2) \\ &= y_2\,(f'-g\,p_A(d_0\,|\,n=m)) \\ &\quad - h\,(1-\tfrac{y_2}{m})\,p_A(d_0\,|\,n=m)\int\left(\sum_{x=0}^{m}(m-x)\,p_A(x\,|\,m,\theta)\right)p_A(\theta)\,\mathrm{d}\theta,\end{aligned} \tag{4.8}$$

with $p_A(d_0\,|\,m)$ defined as in Tactic T1.

The defender does not know the attacker's loss function nor probabilities, so she may assume uncertainty about them and look for the random optimal attack $Y_2^*(m)$ minimising in y_2

$$\begin{aligned}&y_2\,(F'-G\,P_A(d_0\,|\,n=m)) \\ &-H\,(1-\tfrac{y_2}{m})\,P_A(d_0\,|\,n=m)\int\left(\sum_{x=0}^{m}(m-x)\,P_A(x\,|\,m,\theta)\right)P_A(\theta)\,\mathrm{d}\theta,\end{aligned}$$

where F' would be the distribution over cost f'. She would then approximate

$$\hat{p}_D(y_2=0\,|\,m) \approx \#\{Y_{2,k}^*(m)=0\}/K,$$

where $\{Y_{2,k}^*(m)\}_{k=1}^K$ is a sample of size K from $Y_2^*(m)$ drawn in a similar manner to Tactic T1. Note that due to the linearity of the attacker's loss function (4.8), the random optimal attack will be 0 or m, depending on whether it is worth modifying items or not. Non-linear loss functions for the attacker would allow other attacks to take place.

Assumptions about the attacker's random losses and probabilities would be similar to those for Tactic T1. In particular, $F' \sim \mathcal{U}(f_1', f_2')$.

T3: Combination of Tactics T1 and T2

The attacker combines both previous tactics, so he adds y_1 faulty items and converts y_2 of the original items into faults of his. The batch received by the defender consists

of $x - y_2^0$ acceptable items, $m - x - y_2^1$ O-faults and $y_1 + y_2$ A-faults, where y_2^0 and y_2^1 are subject to the restrictions in Tactic T2. The attacker needs to decide y_1 and y_2, which are random to the defender.

Initially, consider that the defender knows the joint distribution $p_D(y_1, y_2 \,|\, m)$. The loss structure for her is as in Table 4.6, with $q_1(n \,|\, \gamma)$ substituted by $q_3(n \,|\, \gamma)$, characterised next. First, the probability of having a final batch of $n = m + y_1$ items, given γ, is

$$p_3(n \,|\, \gamma) = p_1(n \,|\, \gamma),$$

showing the possible initial batch sizes and the included faulty items (as in Tactic T1). Then, the probability that all those items are acceptable ($x = m$ and $y_1 = y_2 = 0$) is

$$q_3(n \,|\, \gamma) = \frac{p_D(y_1 = 0, y_2 = 0 \,|\, m = n)\, p_D(m = n \,|\, \gamma)}{p_3(n \,|\, \gamma)}\, \theta^n, \tag{4.9}$$

since the only combination for an acceptable final batch is having n initial acceptable items ($x = m = n$) and no faulty items included ($y_1 = y_2 = 0$). As for Tactic T1, γ provides information about m.

The expected losses of decisions d_0 (accept) and d_1 (reject) are

$$\psi_D(d_0) = 1 - E_{\theta,\gamma}\left[q_3(n \,|\, \gamma)\right], \qquad \psi_D(d_1) = c\, E_{\theta,\gamma}\left[q_3(n \,|\, \gamma)\right].$$

So the rule is to accept the batch (d_0) if and only if

$$E_{\theta,\gamma}\left[q_3(n \,|\, \gamma)\right] \geq \frac{1}{1+c}, \tag{4.10}$$

which would require simulation to be ascertained.

An ARA scheme is now developed to estimate the quantities $p_D(y_1, y_2 \,|\, m)$ and, thus, $q_3(n \,|\, \gamma)$. To do so, deem the attacker's loss function to be that in Table 4.9, subordinate to the batch composition and the decision made by the defender, as well as his own action. It holds that $x \in \{0, 1, \ldots, m\}$, $y_1 \in \{0, 1, \ldots\}$ and $y_2 = y_2^0 + y_2^1 \in \{0, 1, \ldots, m\}$.

		Final Batch Composition			
		Acceptable	**O-Fault**	**A-Fault**	
				Injected	**Modified**
		$\boldsymbol{x - y_2^0}$	$\boldsymbol{m - x - y_2^1}$	$\boldsymbol{y_1}$	$\boldsymbol{y_2}$
D's Decision	**Accept, d_0**	0	$-h$	$f - g$	$f' - g$
	Reject, d_1	0	0	f	f'

Table 4.9: Attacker's loss per item (T3)

As established in Tactic $T2$, the attacker picks the items to be modified randomly, so that $E\,[y_2^1] = y_2\,\frac{m-x}{m}$. Thus, the attacker's (expected) losses associated with both defender's decisions, when he chooses (y_1, y_2), are

$$l_A(d_0, y_2) = -h\,(m-x)\,(1-\tfrac{y_2}{m}) + (f-g)\,y_1 + (f'-g)\,y_2, \qquad l_A(d_1, y_2) = f\,y_1 + f'\,y_2.$$

The attacker selects (y_1, y_2) to minimise his expected loss when the original size of the batch is m, which is

$$\begin{aligned}\Psi_A(y_1, y_2 \,|\, m) &= p_A(d_0 \,|\, n = m + y_1) \int \left(\sum_{x=0}^{m} l_A(d_0, y_1, y_2)\, p_A(x \,|\, m, \theta) \right) p_A(\theta)\, \mathrm{d}\theta \\ &\quad + (1 - p_A(d_0 \,|\, n = m + y_1))\, l_A(d_1, y_1, y_2) \\ &= y_1\,(f - g\,p_A(d_0 \,|\, n = m + y_1)) + y_2\,(f' - g\,p_A(d_0 \,|\, n = m + y_1)) \\ &\quad - h\,(1 - \tfrac{y_2}{m})\,p_A(d_0 \,|\, n = m + y_1) \int \left(\sum_{x=0}^{m} (m-x)\, p_A(x \,|\, m, \theta) \right) p_A(\theta)\, \mathrm{d}\theta,\end{aligned}$$

with $p_A(d_0 \,|\, n = m + y_1)$ defined as in Tactics T1 and T2.

Uncertainty about the attacker's loss function and probabilities is assumed, since common knowledge is not available to the defender. She looks for the random optimal attack $(Y_1^*, Y_2^*)(m)$ that minimises in y_1 and y_2

$$\begin{aligned}&y_1\,(F - G\,P_A(d_0 \,|\, n = m + y_1)) + y_2\,(F' - G\,P_A(d_0 \,|\, n = m + y_1)) \\ &- H\,(1 - \tfrac{y_2}{m})\,P_A(d_0 \,|\, n = m + y_1) \int \left(\sum_{x=0}^{m} (m-x)\, P_A(x \,|\, m, \theta) \right) P_A(\theta)\, \mathrm{d}\theta\end{aligned}$$

and then estimates

$$\hat{p}_D(y_1, y_2 \,|\, m) \approx \#\{Y_{1,k}^*(m) = y_1, Y_{2,k}^*(m) = y_2\}/K,$$

where $\{(Y_{1,k}^*, Y_{2,k}^*)(m)\}_{k=1}^K$ is a sample of size K from $(Y_1^*, Y_2^*)(m)$ obtained in a similar way as for the previous tactics.

Concerning the attacker's random losses and probabilities, analogous assumptions to those for Tactics T1 and T2 would be made.

4.3.3 Batch acceptance: A numerical example

As an illustration, a numerical example of the analysis in Section 4.3.2 involving Tactic T3 is provided. With regard to the defender's problem, embracing the setting in Section 4.3.1, the elements involved are:

- The rate γ of original incoming items. The prior over γ will be a $\mathcal{G}a(5,1)$ distribution; i.e. the expected size of the original batch will be 5.

- The probability θ that an item is acceptable. The prior over θ will be a $\mathcal{B}e(9,1)$ distribution; i.e. the expected probability of an item's acceptability will be 0.9.

- The (expected) opportunity costs associated with rejecting a batch with all acceptable items will be $c = 0.9$.

As for the attacker's problem, in accordance with assumptions in Section 4.3.2, suppose the following assessments are made:

- The gains and costs will be uniformly distributed as: $F \sim \mathcal{U}(0.25, 0.5)$, $F' \sim \mathcal{U}(0.3, 0.6)$, $G \sim \mathcal{U}(0.8, 1)$ and $H \sim \mathcal{U}(0, 0.25)$. Two implicit assumptions are: (i) on average, injecting A-faults involves less effort for the attacker than modifying items to A-faults as he has broader control over the process (F vs F'); and (ii) the expected gain due to A-faults is greater than that due to O-faults as he may better design them to fulfil his objectives (G vs H).

- $P_A(d_0 \,|\, n)$ will be modelled through a uniform distribution dependent on the final batch size n. To avoid further recursion, consider that the attacker relates it to the defender's non-adversarial version of the problem in Section 4.3.1. In terms of the batch's original expected acceptability, he could expect the the defender to accept it with probability $E_\theta[\theta^n]$. Additionally, he could weigh that probability by 0.5, admitting that the defender might presume him to be manipulating every other batch. In this manner

$$E\left[P_A(d_0 \,|\, n)\right] = \frac{E_\theta[\theta^n]}{2} = \frac{1}{2}\prod_{k=0}^{n-1}\frac{9+k}{10+k} = \frac{9}{18+2n}$$

  is estimated, making use of the defender's prior over θ and expression (4.7). To allow for some uncertainty, and assuming that $P_A(d_0 \,|\, n) > P_A(d_0 \,|\, n+1)$ for every value of n, finally adopt

$$P_A(d_0 \,|\, n) \sim \mathcal{U}\left(\frac{9}{19+2n}, \frac{9\left(1+\frac{1}{9+n}\right)}{19+2n}\right).$$

- Due to its specificity, $P_A(x \,|\, m, \theta)$ will coincide with the $\mathcal{B}in(m, \theta)$ model of the defender.

- $P_A(\theta)$ will be a Dirichlet process with a Beta distribution base $\mathcal{B}e(9,1)$ and concentration parameter $\rho = 100$.

Recall that $y_2^* \in \{0, m\}$. Therefore, the attack probabilities for each original batch size m may be estimated as follows:

Algorithm 6 Batch acceptance (T3): Simulation of attack probabilities

Data: Original batch size m; number of iterations K; upper bound for the amount of injected items $\overline{Y}_1$.

1: Set $p(y_1, y_2) = 0, \quad y_1 = 0, \ldots, \overline{Y}_1,\ y_2 = 0, m.$

2: **For** $k = 1$ **to** K **do**

3: Generate $f_k \sim \mathcal{U}(0.25, 0.50)$.

4: Generate $f'_k \sim \mathcal{U}(0.30, 0.60)$.

5: Generate $g_k \sim \mathcal{U}(0.80, 1.00)$.

6: Generate $h_k \sim \mathcal{U}(0.00, 0.25)$.

7: Generate distribution $p_A^k(\theta) \sim \mathcal{D}ir\mathcal{P}(\mathcal{B}e(9, 1), 100)$.

8: **For** $y_1 = 0$ **to** $\overline{Y}_1$ **do**

9: Generate $\pi_A^{0,k}(y_1) \sim \mathcal{U}\left(\dfrac{9}{19 + 2m + 2y_1}, \dfrac{9\left(1 + \frac{1}{9+m+y_1}\right)}{19 + 2m + 2y_1}\right)$.

10: $\psi_A^k(y_1, 0) = y_1 \left(f_k - g_k\, \pi_A^{0,k}(y_1)\right)$
$- h_k\, \pi_A^{0,k}(y_1) \displaystyle\int \left(\sum_{x=0}^{m} \binom{m}{x} \theta^x\, (1-\theta)^{m-x}\, (m - x)\right) p_A^k(\theta)\, \mathrm{d}\theta.$

11: $\psi_A^k(y_1, m) = y_1 \left(f_k - g_k\, \pi_A^{0,k}(y_1)\right) + m \left(f'_k - g_k\, \pi_A^{0,k}(y_1)\right).$

12: **End For**

13: Find $(y_1^*, y_2^*) = \underset{y_1, y_2}{\arg\min}\ \psi_A^k(y_1, y_2)$.

14: Set $p(y_1^*, y_2^*) = p_(y_1^*, y_2^*) + 1$.

15: **End For**

16: Set $\hat{p}_D(y_1^*, y_2^*) = p(y_1^*, y_2^*)/K, \quad y_1 = 0, \ldots, \overline{Y}_1,\ y_2 = 0, m.$

Algorithm 6 has been coded in `R` (Appendix B.4). Tables 4.10 and 4.11 reflect an application of the previous scheme with $K = 10^3$ iterations (sufficient for illustrative purposes as the procedure is computationally intensive due to the need to sample from the Dirichlet process) and an upper bound for the amount of injected items of $\overline{Y}_1 = 4$, leading to the estimates of $\hat{p}_D(y_1, y_2 \,|\, m)$ for an original batch size of $m = 0, 1, \ldots, 8$.

		Original Batch Size - m								
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Attack $\bar{y}_1$	**0**	0.374	0.375	0.557	0.726	0.806	0.902	0.966	0.995	1.000
	1	0.292	0.183	0.157	0.142	0.133	0.086	0.034	0.005	0.000
	2	0.203	0.129	0.103	0.054	0.028	0.009	0.000	0.000	0.000
	3	0.104	0.048	0.025	0.010	0.001	0.000	0.000	0.000	0.000
	4	0.027	0.005	0.002	0.001	0.000	0.000	0.000	0.000	0.000

Table 4.10: Defender's estimation of $\hat{p}_D(y_1, y_2 \,|\, m)$ with $y_2 = 0$

		Original Batch Size - m								
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Attack $\bar{y}_1$	**0**	–	0.200	0.142	0.064	0.032	0.003	0.000	0.000	0.000
	1	–	0.044	0.013	0.003	0.000	0.000	0.000	0.000	0.000
	2	–	0.015	0.001	0.000	0.000	0.000	0.000	0.000	0.000
	3	–	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	4	–	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 4.11: Defender's estimation of $\hat{p}_D(y_1, y_2 \,|\, m)$ with $y_2 = m$

Plugging such values in (4.9), the defender may compute the expected probability that all items are acceptable conditional on the observed size of the final batch. In Table 4.12, those values are provided, as well as the defender's optimal decision based on expression (4.10), with threshold $1/(1+c) = 0.526$. According to Tables 4.10 and 4.11, the final batch size possibilities that may be encountered are just $n = 0, 1, \ldots, 8$.

The following remarks may be deduced from Tables 4.10, 4.11 and 4.12:

- When an empty batch is received ($n = 0$), the model behaves correctly and accepts the batch since the defender knows that there are no faulty items.

	Final Batch Size - n								
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
$\boldsymbol{E_{\theta,\gamma}}\left[\boldsymbol{q_3(n \mid \gamma)}\right]$	1.000	0.523	0.546	0.555	0.516	0.498	0.493	0.511	0.524
Accept, d_0	Yes	No	Yes	Yes	No	No	No	No	No

Table 4.12: Defender's optimal decision given a final batch of n items

- For smaller original batch sizes, the attacker is encouraged to both inject and/or modify items as it is more likely that all original items are acceptable. This might cause the defender not to accept batches with a small final size ($n = 1$ in the current example).
- For bigger original batch sizes, the attacker is discouraged to intervene and thus avoid costs as it is more likely that some original items are already faulty. This might lead the defender to accept batches with a medium final size ($n = 2, 3$ in the ongoing example).
- When a sufficiently large batch is received (starting with $n = 4$ in the provided example), the defender will not accept the batch as she will expect the original one to formerly include faulty items.

4.4 Recapitulation

The AHT problem has been examined in detail from an ARA perspective. In this manner, concept uncertainty and common knowledge limitations frequently found in game-theoretic solutions to adversarial classification and adversarial machine learning have been avoided. Support has been provided to a decision-maker who essentially needs to ascertain which of several hypotheses holds, based on observations from a source that may be perturbed by another agent with some purpose. The main focus has been on testing two simple hypothesis, but the framework extends to other types of hypothesis tests.

Specific models for explicit AHT problems have been developed and numerical examples have been provided and implemented in `R`, with the code included in Appendices B.3 and B.4. These differ in the degree of information perceived by each of the agents and, in particular, an elaborate batch acceptance model has been devised to be applied to e.g. spam detection or container screening at international ports. Further real-world applications to adversarial classification may be found in Naveiro et al. [2018].

Chapter 5

Conclusions

ARA is an emergent paradigm in the context of strategic reasoning in conflict situations and, as such, much work remains to be completed so as to spread its use and make it widely known. This chapter recaps the PhD thesis developments in Chapters 2, 3 and 4 and proposes new problems to extend that work, respectively, in Sections 5.1, 5.2 and 5.3. Besides, Section 5.4 suggests applying an ARA perspective to (stochastic) differential games to transcend discrete decision-making processes and approach continuous ones.

These papers with the most significant results of this PhD thesis have already been published:

González-Ortega, J.; Ríos Insua, D. & Cano, J. (2018). Adversarial risk analysis for bi-agent influence diagrams: An algorithmic approach. *European Journal of Operational Research*, in press, `https://doi.org/10.1016/j.ejor.2018.09.015`.

Ríos Insua, D.; Banks, D.L.; Ríos, J. & González-Ortega, J. (2017). Adversarial risk analysis. In *Wiley StatsRef: Statistics Reference Online*, `https://doi.org/10.1002/9781118445112.stat07972`.

Ríos Insua, D.; González-Ortega, J.; Banks, D.L. & Ríos, J. (2018). Concept uncertainty in adversarial statistical decision theory. In *The Mathematics of the Uncertain: A Tribute to Pedro Gil*, 2018 ed., Springer, Cham, Switzerland, 527–542.

While the next still awaits to be published:

González-Ortega, J.; Ríos Insua, D.; Ruggeri, F. & Soyer, R. (exp. 2019). Hypothesis testing in presence of adversaries. Submitted to *The American Statis-*

tician.

Besides, a number of talks have also been derived from its contents:

González-Ortega, J. (May 26th, 2016). Adversarial risk analysis for bi-agent influence diagrams. Oral presentation at the *Workshop on Adversarial Risk Analysis for Critical Infrastructure*, LC – NIAS, Leiden, Netherlands.

González-Ortega, J. (September 5th, 2016). Adversarial risk analysis for bi-agent influence diagrams. Oral presentation at the *36th Spanish Conference on Statistics and Operational Research*, SEIO – UCLM, Toledo, Spain.

González-Ortega, J. (September 23rd, 2016). Adversarial risk analysis. Oral presentation at the *10th Workshop of Young Researchers in Mathematics*, UCM, Madrid, Spain.

González-Ortega, J. (December 12th, 2016). Adversarial hypothesis testing. Poster presentation at the *SRA 2016 Annual Meeting*, SRA, San Diego, CA.

González-Ortega, J. (June 7th, 2017). Adversarial risk analysis for bi-agent influence diagrams. Oral presentation at the *5th Symposium on Games and Decisions in Reliability and Risk*, ICMAT – RAC, Madrid, Spain.

González-Ortega, J. (June 27th, 2017). Adversarial risk analysis for generic agent interactions. Oral presentation at the *Advances in Decision Analysis 2017 Conference*, DAS – UT, Austin, TX.

González-Ortega, J. (October 13th, 2017). Hypothesis testing in presence of adversaries. Oral presentation at the *9th Workshop on Dynamic Games in Management Science*, GERAD – UdeM, Montréal, Canada.

González-Ortega, J. (December 11th, 2017). An algorithmic adversarial risk analysis approach for bi-agent influence diagrams. Poster presentation at the *SRA 2017 Annual Meeting*, SRA, Arlington, VA.

González-Ortega, J. (June 18th, 2018). Adversarial statistical decision theory: Embracing adversarial risk analysis. Oral presentation at the *SRA-E 2018 Annual Conference*, SRA-E – MiUn, Östersund, Sweden.

González-Ortega, J. (June 28th, 2018). Adversarial statistical decision theory. Oral presentation at the *2018 ISBA World Meeting*, ISBA – UoE, Edinburgh, UK.

Additional related work to that of this PhD thesis has been conducted in:

González-Ortega, J.; Radovic, V. & Ríos Insua, D. (2018). Utility elici-

tation. In *Elicitation: The Science and Art of Structuring Judgement*, 2018 ed., Springer, Cham, Switzerland, 241–264.

Gómez Esteban, P.; Liu, A.; Ríos Insua, D. & González-Ortega, J. (exp. 2019). Competition and cooperation in a community of autonomous agents. Submitted to *Autonomous Robots*.

Ríos Insua, D.; Banks, D.L.; Ríos, J. & González-Ortega, J. (exp. 2019). Structured expert judgement modelling through adversarial risk analysis. Forthcoming.

5.1 Bi-agent influence diagrams

Objective O1 proposed developing representation methods in ARA, which has been approached by producing an ARA algorithmic scheme to evaluate proper BAIDs and applying it to solve a simplified CIP problem. Results in Chapter 2 have been condensed into a paper on an ARA algorithmic evaluation of BAIDs which is to be published in EJOR [González-Ortega, Ríos Insua and Cano, 2018] and has already been presented in several international conferences. Additionally, the code developed to compute BAIDs, available in Appendix B.1, is intended to be disclosed in an `R` public package.

Of the different elements which the decision-maker uses to model her uncertainty about the opponent's probabilities and utilities, those concerning chance BAID nodes are relatively standard decision-analytic assessments. They rely on the supported agent's beliefs about her adversary's judgements over the results of their interaction, which she can always base on her own probability distributions with some uncertainty around it. Usual candidates would be a Dirichlet distribution in the discrete case or a Dirichlet process in the continuous one, as they constitute probability distributions whose domains are themselves sets of probability distributions. Banks et al. [2015] provide further examples from those in this PhD thesis of the use of Dirichlet distributions and processes in the modelling of probability distributions with uncertainty. As far as utility nodes are concerned, adversarial utility elicitation has been extensively reviewed in González-Ortega, Radovic and Ríos Insua [2018]. Further information may be found in Ríos Insua et al. [exp. 2019].

However, the decision-maker's own decisions as perceived by the opponent entail strategic thinking and could lead to recursions when higher level-k thinking strategies are considered, as discussed in Ríos and Ríos Insua [2012] for a simpler class of problems. Algorithms in this PhD thesis could be extended to consider higher level-k strategies, though, in any case, the development of additional procedures to the Beta and Dirichlet distributions used in the example poses a challenge that

should be undertaken as a future line of research.

An effort should be made to promote efficient computational schemes for BAIDs. One possibility would be to try a single stage process based on the augmented probability simulation approach to ID valuation [Bielza et al., 1999]. Another possibility would be to use loop parallelisation algorithms [Boulet et al., 1998]. It might be interesting to produce a computational environment supporting this methodology, possibly linked with or embedded within GeNIe [1998].

BAIDs, similarly to IDs, symmetrise problems inducing inefficiencies in some cases as in Bielza and Shenoy [1999]. This issue becomes manifest in sequential decision diagrams [Call and Miller, 1990], sequential valuation networks [Demirer and Shenoy, 2006], multi-agent versions of decision circuits [Bhattacharjya and Shachter, 2012] or chain event graphs [Thwaites and Smith, 2017]. All these alternative representation methods could be explored under an ARA perspective.

Multiple attacker and/or multiple defender cases are also of interest. An ARA approach of MAIDs would support one of the agents (the defender) against the rest (the attackers), see Hausken and Bier [2011] for a game-theoretic framework. In this case, it would be necessary to differentiate the possibilities in which attackers are completely independent, partially or totally coordinated or such that their attacks influence somehow each other. It could also be the case that there are various defenders, possibly cooperating.

Finally, the ARA algorithmic approach to BAIDs has been illustrated with a CIP example considering a simplified version of a single installation structure. However, if the model were to be applied to a real case, the attacker's choice contemplated in the model on whether to infiltrate the system would substantially condition his final decision on whether to attack and, thus, a discussion on the value of information and secrecy would be relevant, see Zhuang and Bier [2010], Ríos and Ríos Insua [2012] and Zhang et al. [2015]. Moreover, a multi-period problem which would allow for information updating, as in Zhuang et al. [2010], could be considered. In addition, the figure of merit would need to be reassessed as it is over-simplified to the number of days of service shortage. Other criteria that could be recognised are the repair costs, the number of captured terrorists or the number of fatalities if it were a violent attack. Furthermore, the attacker's interests may reside in different features than those of the defender, e.g. Keeney [2007] and Keeney and von Winterfeldt [2010] provide what may be viewed as catalogues in the domain of terrorism. The simplification was made for the sake of a better understanding of the numerical example and a reduction of the problem's size and computation time.

5.2 Concept uncertainty

The enhancement of the modelling of strategic reasoning was the aim of Objective O2 and, to that end, concept uncertainty has been explored within an ARA perspective using ASDT to procure an example in adversarial point estimation. An online reference of ARA [Ríos Insua et al., 2017] and a paper [Ríos Insua et al., 2018] have already been published covering the contents of Chapter 3. However, just two different solution concepts have been considered in this PhD thesis. Additional classes of adversaries that could be examined would include non-strategic or prospect-maximising players, among others. The model requires assessing the weights of various solution concepts, which would be done based on expert judgement. Should the situation be repeated over time, a strategy for learning about the relevance solution concepts based on a Bayesian updating scheme could be introduced, as suggested in Ríos and Ríos Insua [2012].

Game theory and ARA are different frameworks to provide support to decision-makers in competitive situations. As such, they differ in their core assumptions about the available information, shared knowledge and consequent solution concept and, indeed, lead to different solutions as shown in Chapters 3 and 4 or in e.g. Ríos Insua, Ruggeri et al. [2016]. The prescribed decisions for the decision-maker provided by both methodologies might coincide or not, but the underlying rationale for any enacting player would be different, as well as the information each of them handles about the other(s). Future work could systematically show the connection and relation between both types of solutions, possibly in line with the approach in Gómez Esteban et al. [exp. 2019].

5.3 Adversarial hypothesis testing

Research on the AHT problem was specially conceived to comply with transversal Objective O3 on the implementation of ARA models. Towards that end, a paper [González-Ortega et al., exp. 2019] has been produced and, though it still awaits publication, it has already been recognised with the Student Merit Award of the Security & Defense Specialty Group (SDSG) in the SRA 2016 Annual Meeting.

An illustrative application in relation with batch acceptance has been studied. However, it was assumed that the decision-maker only observed the batch size, which might not be the case (e.g. when screening containers at international ports) so that supplementary partial information could be considered. For example, it could be the case that, besides the batch size, she observes some item features and this information is incorporated to the hypothesis testing problem. Moreover, when the decision-maker has no information about the batch size other than her previous experience, a multi-stage version of the model could be proposed, maybe considering

hidden states as in Zhang and Zenios [2008]. New strategies for the adversary such as the injection of (apparently) acceptable items to confound the decision-maker could be considered. Other loss functions could be explored as well.

Additional applications may be found in the context of, for example, adversarial signal processing, such as in Electronic Warfare (EW) where pulse/signal environment is generally very complex with many different radars transmitting simultaneously. The time interval between two pulses emitted by a threat radar is defined as the Pulse Repetition Interval (PRI). PRI tracking is an important problem in naval EW applications because knowledge of the PRI is used to defend ships against radar-guided missiles. The signals received may be jammed by hostile radars and this results in missing pulses due to reduced sensitivity of the receiver, see Hock and Soyer [2006] for an introduction.

5.4 (Stochastic) differential games

This PhD thesis, and more generally the research in ARA, has focused on decision-making processes in which the various agents' choices are discrete in time. However, there are many situations such that the strategies determine courses of action over time. Thus, the ARA methodology should be extended so as to be applicable to continuous non-cooperative decision-making processes.

Among them, conflicts related to dynamical systems, known as Differential Games (DGs), present a relevant subset of problems that could be first approached. DGs are mathematical models describing the strategic interactions between two or more agents who affect the evolution of a system over time. Its applications are significant, for example, in economic competition, security conflicts, disease control, predator-prey relations or environmental regulation. Their methods may be seen as a combination of game theory and optimal control theory and have their origin in the work of Isaacs [1965]. Friedman [1972] introduced stochastic elements in the evolution of the system leading to the so-called Stochastic Differential Games (SDGs). A recent review may be seen in the updated 2015 edition of Nisio [1981]. Optimal strategies for SDGs based on ARA could be elaborated, performing inference and prediction on stochastic processes while acknowledging the presence of an underlying adversary.

Essentially, DGs consider that the evolution of a system's state in time is dictated by a set of differential equations that depend on the state itself and the decisions made by two or more agents, who receive payoffs conditioned on the evolution of the system, the final state and the implemented actions. As a fundamental solution concept, Nash equilibria are computed as, for example, in Basar and Li [1989]. This requires that each agent knows the status of the others, which is hardly sustainable

in many applications and motivates the use of ARA. To calculate and prove the existence of the equilibrium, optimal control concepts such as Pontryagin's maximum principle [Pontryagin, 1961] are employed and enhanced. In SDGs, a known stochastic element (in the sense that the corresponding stochastic process parameters are assumed to be known) is added, which again is difficult to accept in many applications. An ARA perspective for SDGs would avoid the current problem of common knowledge assumptions and, as a Bayesian approach, could introduce uncertainty in the underlying stochastic processes, e.g. as in Ríos Insua et al. [2012] and Gómez-Corral et al. [2015].

Preliminary work, already presented in the SEIO 2018 Spanish Conference on Statistics and Operational Research and the EURO 2018 European Conference on Operational Research, has been conducted for DGs replicating the game-theoretic botnet defence model in Bensoussan et al. [2010]. The undertaken ARA approach transforms the relevant deterministic DG into two stochastic control problems, one for each of the involved agents, in which the stochasticity derives from the opponent's uncertain strategy. Though optimal ARA solutions still need to be developed, results already show that deviating from the proposed Nash equilibria and reflecting the decision-maker's uncertainty about the adversary's chosen strategy may reduce her expected losses. Therefore, an effort should be made towards consolidating an ARA framework for DGs and SDGs.

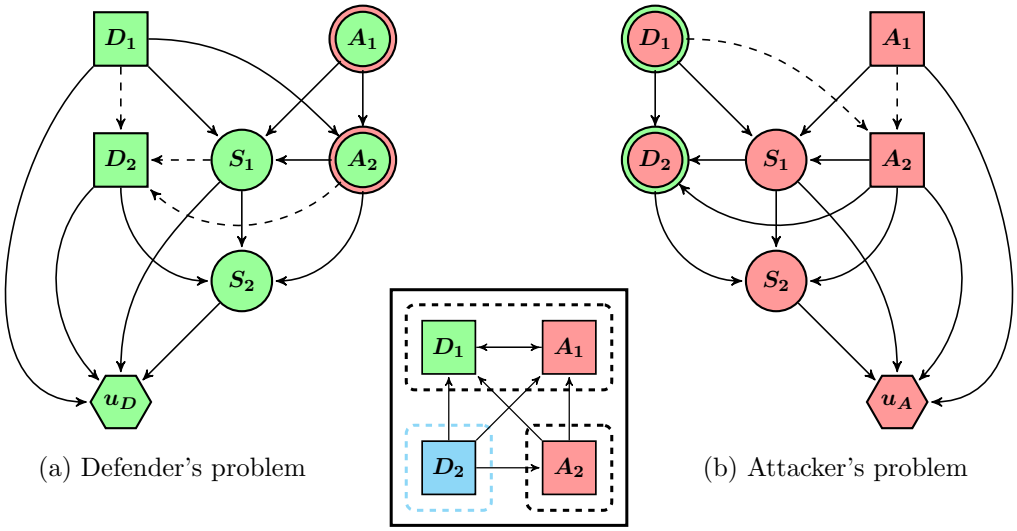
Appendix A

CIP algorithmic solution

In an attempt to better illustrate how the general computational strategy for BAIDs in Chapter 2 works, scheme $\mathcal{G}$ to solve the CIP example is graphically displayed here. For each $\mathcal{G}$ step, the relevant problem $\mathcal{D}$ (defender) or $\mathcal{A}$ (attacker) is identified and a decomposition into the corresponding substeps is conducted.

Step $\mathcal{G}1$ ($\mathcal{D}$)

Select decision node D_2 in SCC $\{D_2\}$ to be solved (problem $\mathcal{D}$).



(a) Defender's problem

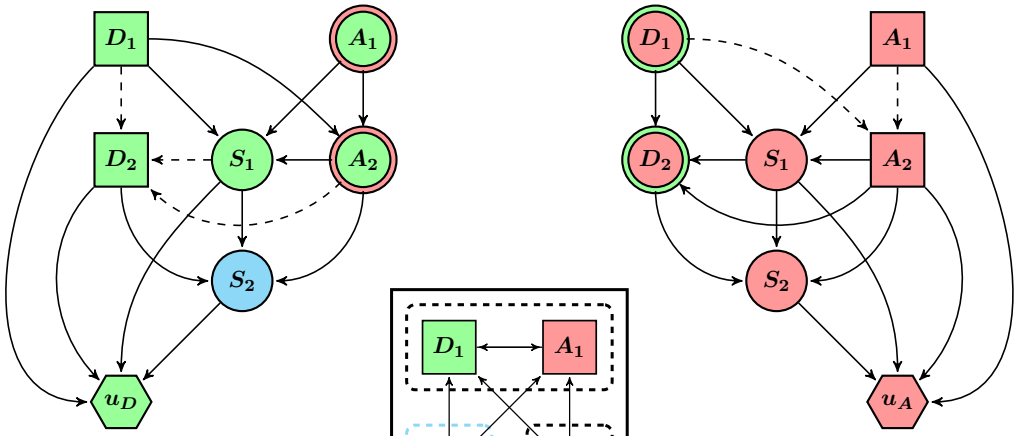
(b) Attacker's problem

(c) Relevance graph

Figure A.1: Step $\mathcal{G}1$ ($\mathcal{D}$)

Step $\mathcal{D}1$

Remove chance node S_2 in $\mathcal{D}$.



(a) Defender's problem

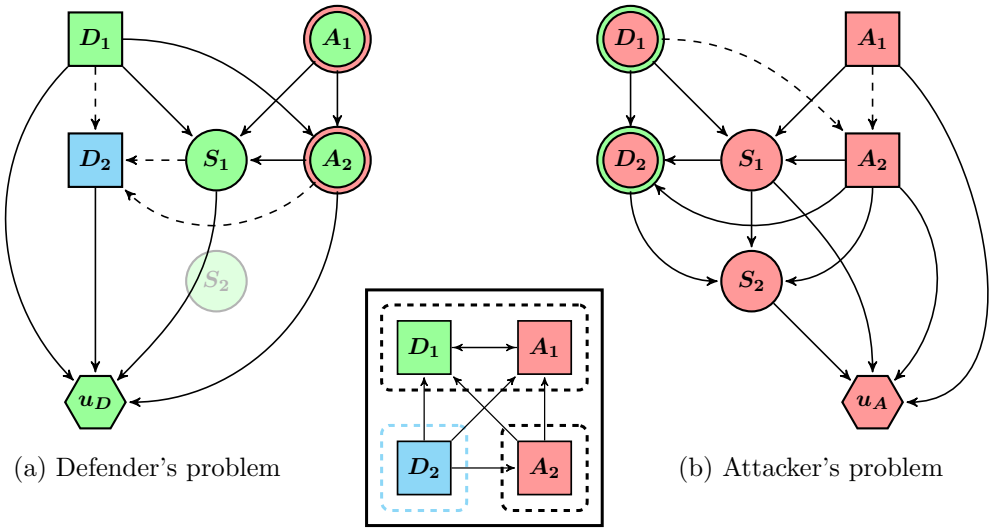
(b) Attacker's problem

(c) Relevance graph

Figure A.2: Step $\mathcal{D}1$

Step $\mathcal{D}2$

Eliminate decision node D_2 in $\mathcal{D}$ and store optimal action $d_2^*(d_1, a_2, s_1)$.



(a) Defender's problem

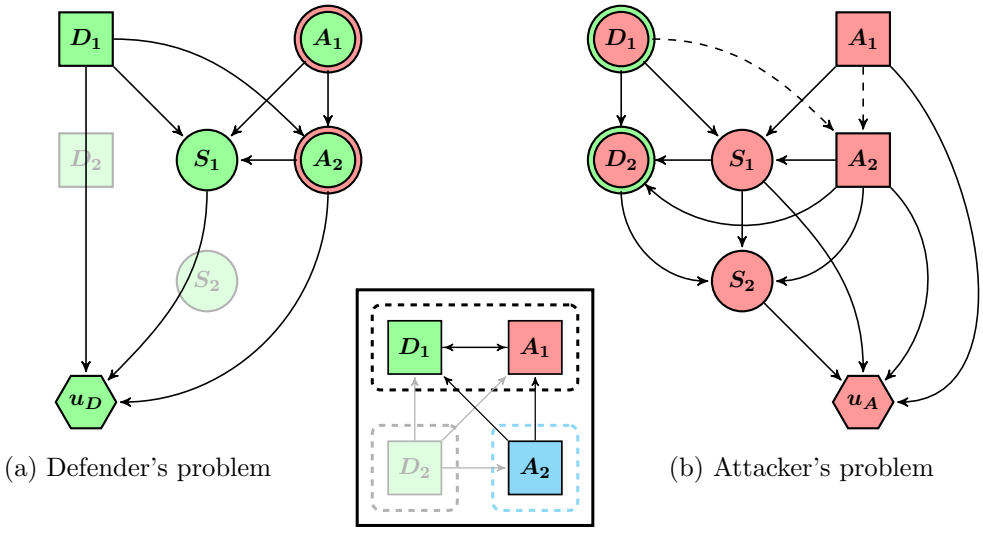
(b) Attacker's problem

(c) Relevance graph

Figure A.3: Step $\mathcal{D}2$

Step $\mathcal{G}2$ ($\mathcal{A}$)

Tag decision node A_2 in SCC $\{A_2\}$ to be solved (problem $\mathcal{A}$).



(a) Defender's problem

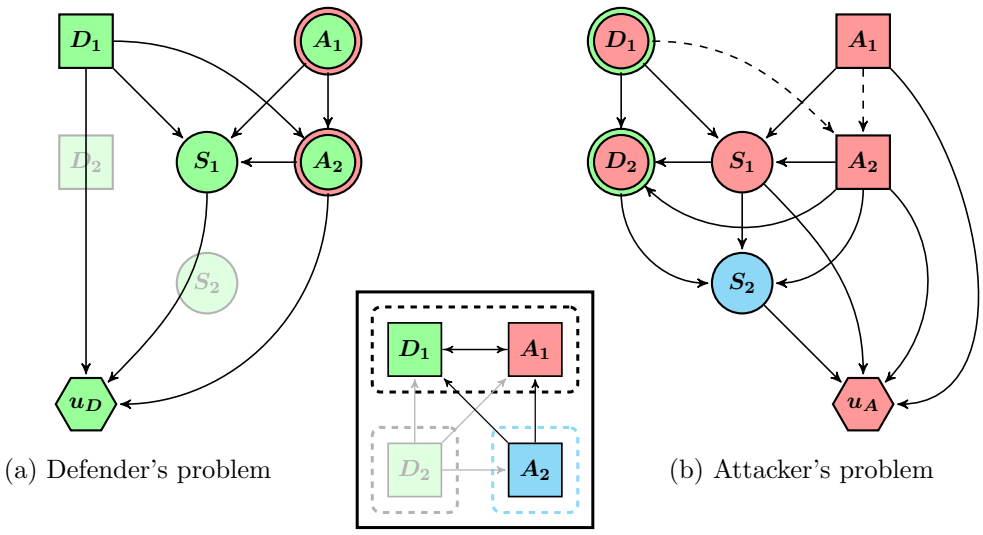
(b) Attacker's problem

(c) Relevance graph

Figure A.4: Step $\mathcal{G}2$ ($\mathcal{A}$)

Step $\mathcal{A}1$

Remove chance node S_2 in $\mathcal{A}$.



(a) Defender's problem

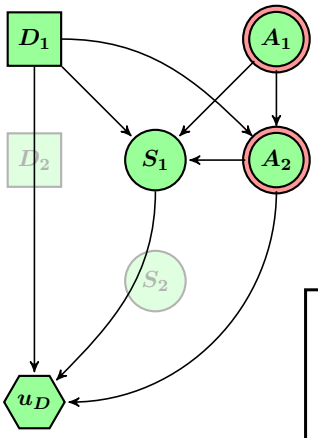
(b) Attacker's problem

(c) Relevance graph

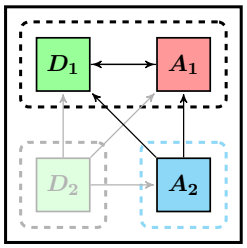
Figure A.5: Step $\mathcal{A}1$

Step $\mathcal{A}2$

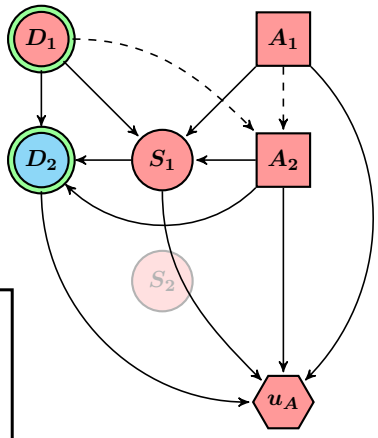
Eliminate chance (for the attacker) node D_2 in $\mathcal{A}$.



(a) Defender's problem



(c) Relevance graph

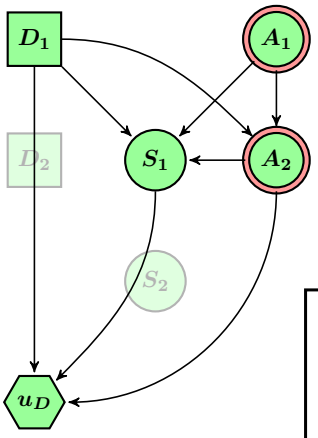


(b) Attacker's problem

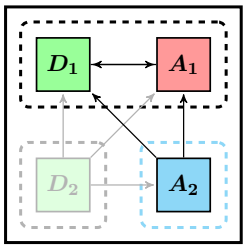
Figure A.6: Step $\mathcal{A}2$

Step $\mathcal{A}3$

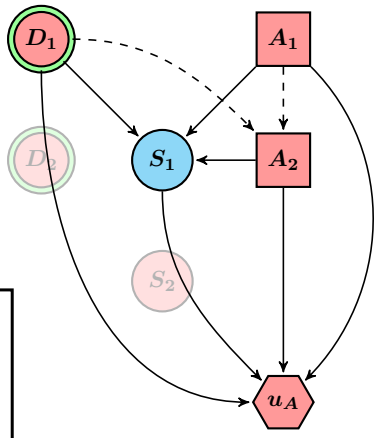
Take out chance node S_1 in $\mathcal{A}$.



(a) Defender's problem



(c) Relevance graph

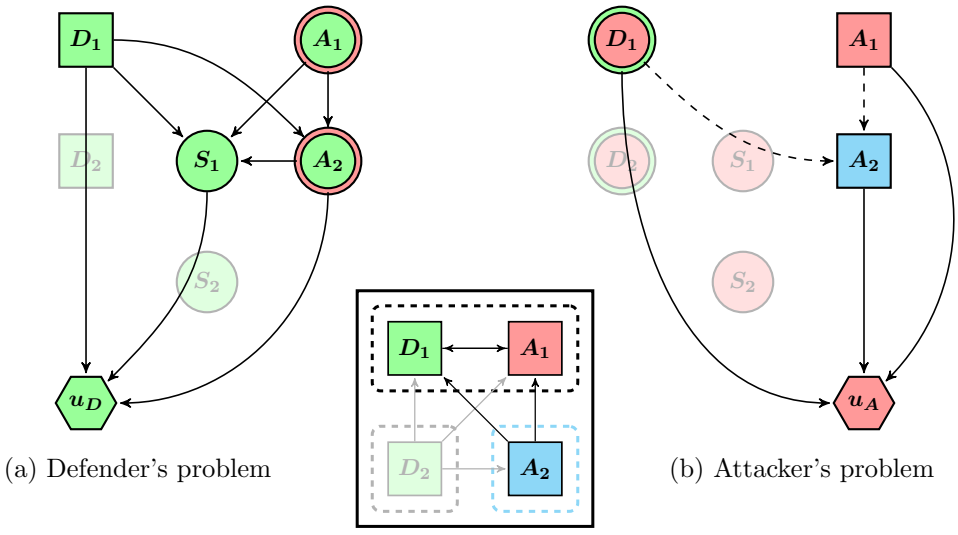


(b) Attacker's problem

Figure A.7: Step $\mathcal{A}3$

Step $\mathcal{A}4$

Withdraw decision node A_2 in $\mathcal{A}$ and record (random) optimal action $A_2^*(d_1, a_1)$.



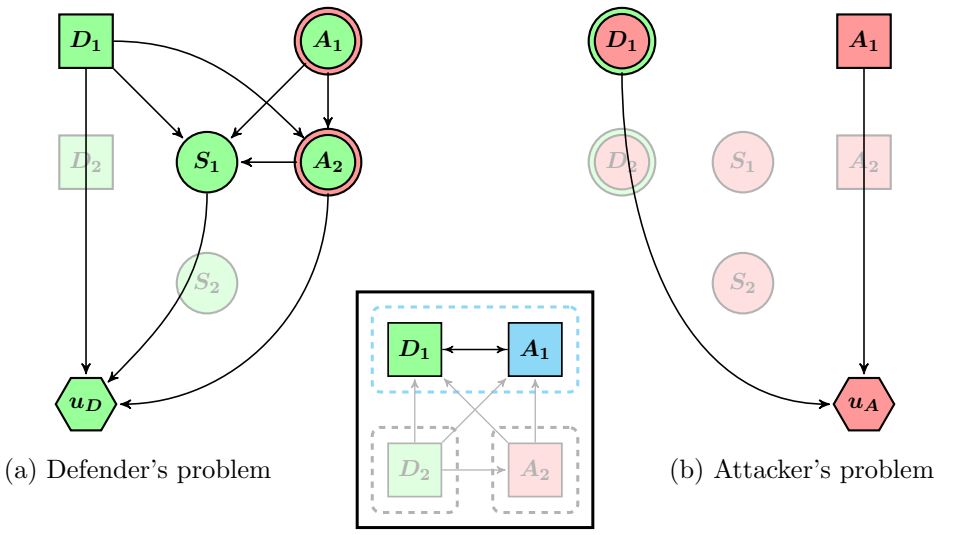
(a) Defender's problem (b) Attacker's problem

(c) Relevance graph

Figure A.8: Step $\mathcal{A}4$

Step $\mathcal{G}3$ ($\mathcal{A}$)

Choose decision node A_1 in SCC $\{D_1, A_1\}$ to be solved (problem $\mathcal{A}$).



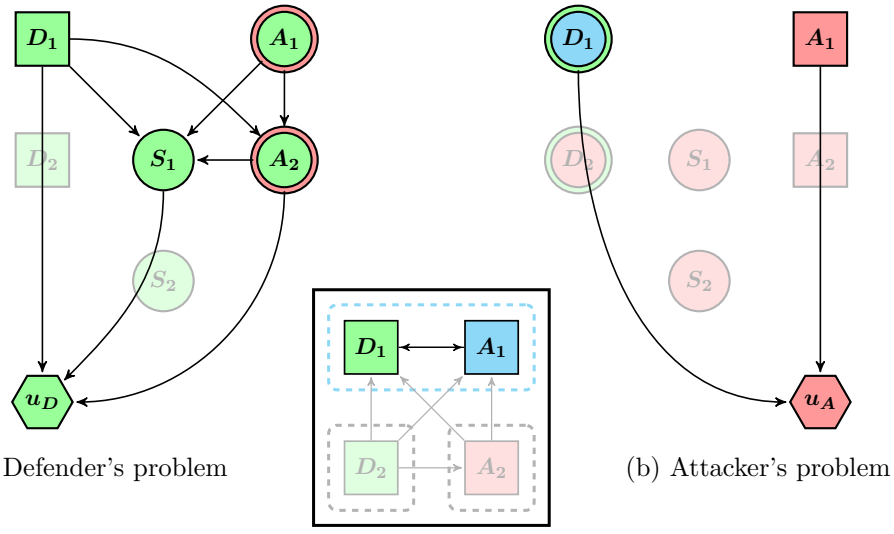
(a) Defender's problem (b) Attacker's problem

(c) Relevance graph

Figure A.9: Step $\mathcal{G}3$ ($\mathcal{A}$)

Step $\mathcal{A}5$

Delete chance (for the attacker) node D_1 in $\mathcal{A}$.



(a) Defender's problem

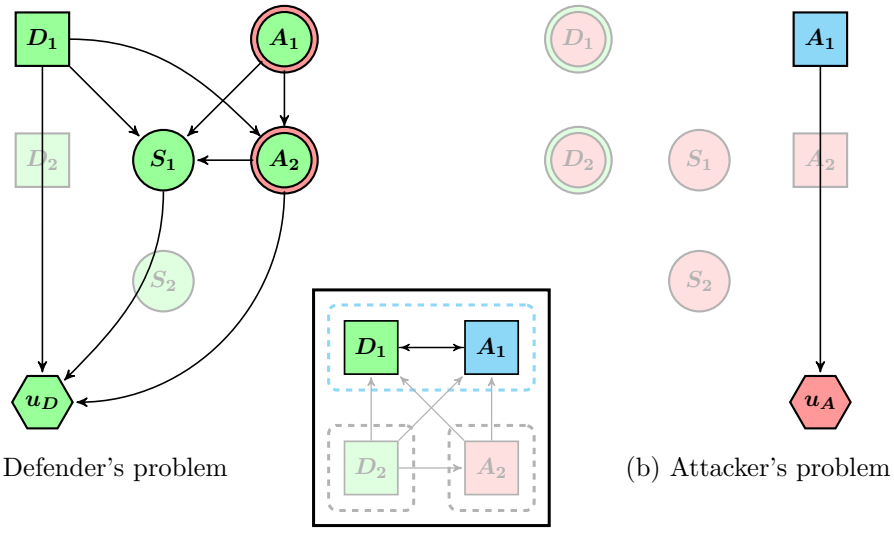
(b) Attacker's problem

(c) Relevance graph

Figure A.10: Step $\mathcal{A}5$

Step $\mathcal{A}6$

Reduce decision node A_1 in $\mathcal{A}$ and save (random) optimal action A_1^*.



(a) Defender's problem

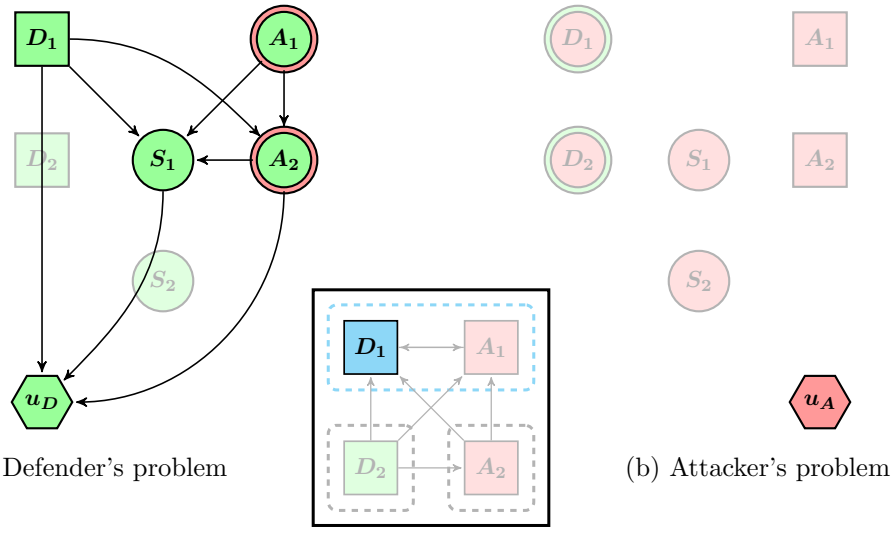
(b) Attacker's problem

(c) Relevance graph

Figure A.11: Step $\mathcal{A}6$

Step $\mathcal{G}4$ ($\mathcal{D}$)

Pick decision node D_1 in SCC $\{D_1, A_1\}$ to be solved (problem $\mathcal{D}$).



(a) Defender's problem

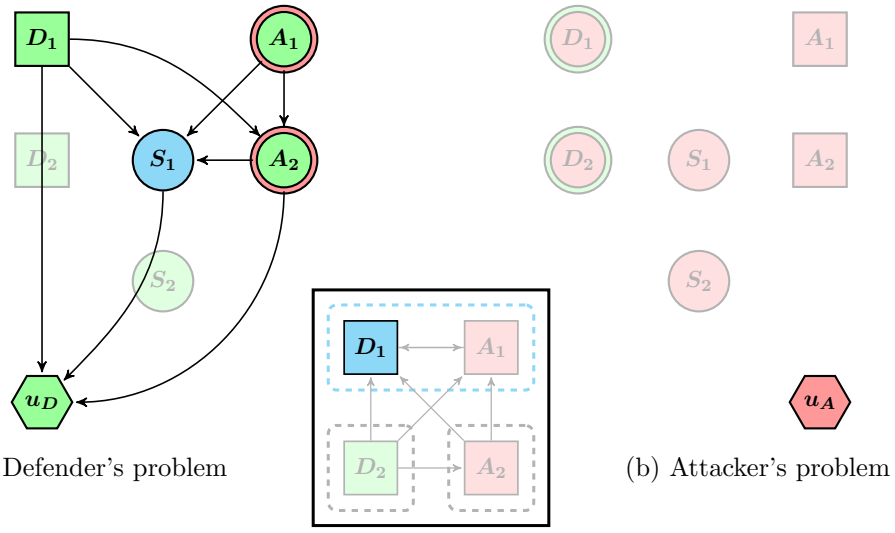
(b) Attacker's problem

(c) Relevance graph

Figure A.12: Step $\mathcal{G}4$ ($\mathcal{D}$)

Step $\mathcal{D}3$

Take out chance node S_1 in $\mathcal{D}$.



(a) Defender's problem

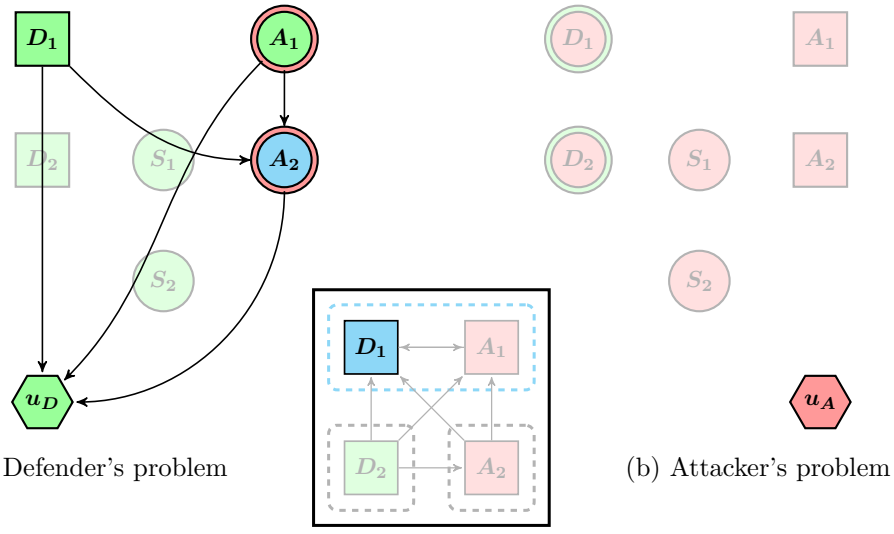
(b) Attacker's problem

(c) Relevance graph

Figure A.13: Step $\mathcal{D}3$

Step $\mathcal{D}4$

Withdraw chance (for the defender) A_2 in $\mathcal{D}$ using $A_2^*(d_1, a_1)$.



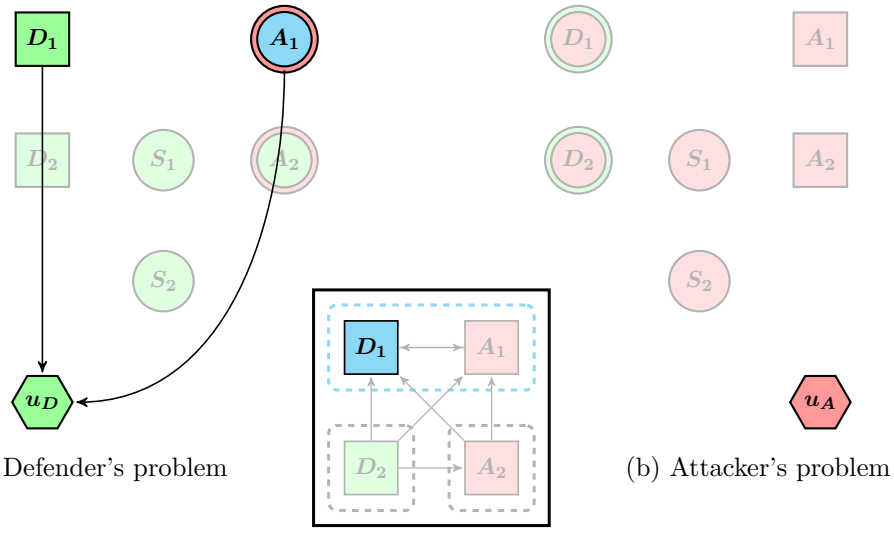
(a) Defender's problem (b) Attacker's problem

(c) Relevance graph

Figure A.14: Step $\mathcal{D}4$

Step $\mathcal{D}5$

Delete chance (for the defender) A_1 in $\mathcal{D}$ employing A_1^*.



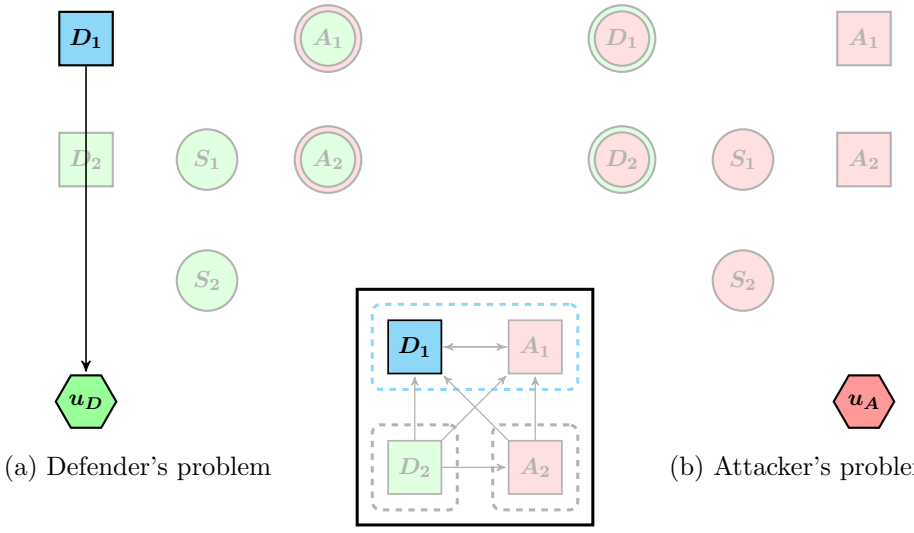
(a) Defender's problem (b) Attacker's problem

(c) Relevance graph

Figure A.15: Step $\mathcal{D}5$

Step $\mathcal{D}6$

Reduce decision node D_1 in $\mathcal{D}$ and find optimal action d_1^*.



(a) Defender's problem

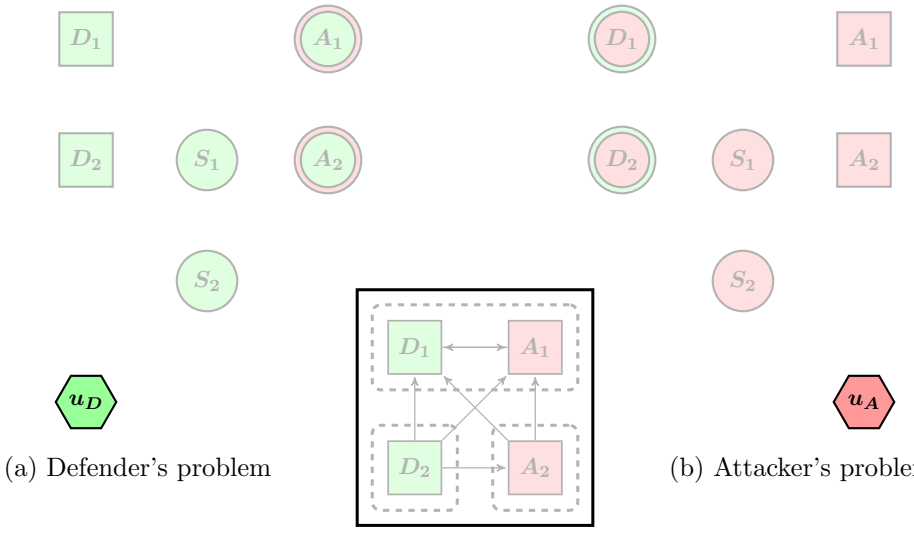
(b) Attacker's problem

(c) Relevance graph

Figure A.16: Step $\mathcal{D}6$

End

The defender should choose d_1^* at D_1 and, later, $d_2^*(d_1, a_2, s_1)$ at D_2.



(a) Defender's problem

(b) Attacker's problem

(c) Relevance graph

Figure A.17: End

Appendix B

Code

The original R and MATLAB codes used for the resolution of the numerical examples in Chapters 2, 3 and 4 is incorporated here.

B.1 Bi-agent influence diagrams (R)

```
## SETTINGS ############################################################

# Required libraries:

library(gtools)

# Monte Carlo iterations:

K <- 10^5

## INPUT ##############################################################

# BAID:

B <- list()

# Number of nodes:

B$n <- 8

# Value nodes (1st row = D / 2nd row = A):

B$V <- matrix(c("7","8"),
              nrow = 2, ncol = 1, byrow = TRUE)

# Decision nodes (1st row = D / 2nd row = A):
```

```
B$D <- matrix(c("1","4","2","5"),
              nrow = 2, ncol = 2, byrow = TRUE)

# Chance nodes:

B$C <- c("3","6")

# Arcs:

B$A <- matrix(0,B$n,B$n)
rownames(B$A) <- c(1:8); colnames(B$A) <- rownames(B$A)

B$A["1","3"] <- 1; B$A["1","4"] <- 1; B$A["1","5"] <- 1
B$A["1","7"] <- 1; B$A["2","3"] <- 1; B$A["2","5"] <- 1
B$A["2","8"] <- 1; B$A["3","4"] <- 1; B$A["3","6"] <- 1
B$A["3","7"] <- 1; B$A["3","8"] <- 1; B$A["4","6"] <- 1
B$A["4","7"] <- 1; B$A["5","3"] <- 1; B$A["5","4"] <- 1
B$A["5","6"] <- 1; B$A["5","8"] <- 1; B$A["6","7"] <- 1
B$A["6","8"] <- 1

# Nodes:
#   type - Continuous (c) / Discrete (d) / Utility (u)
#   set - Value set (interval bounds if c)
#   1 - D's assessment
#   2 - A's assessment

B$f <- list()
for(i in rownames(B$A)){
  B$f[[i]] <- list()}
rm(i)

B$f[["1"]]$type <- "d"
B$f[["1"]]$set <- c(0,1)
B$f[["1"]][["1"]] <- function(y=
                      rep(0,length(which(B$A[,"1"] == 1)))){
  names(y) <- sortN(names(which(B$A[,"1"] == 1)))
  f <- function(x){
    if(x %in% B$f[["1"]]$set){
      return(1)}
    else{
      return(0)}}
    return(f)}
B$f[["1"]][["2"]] <- function(y=
                      rep(0,length(which(B$A[,"1"] == 1)))){
  names(y) <- sortN(names(which(B$A[,"1"] == 1)))
  prob <- array(0, dim = length(B$f[["1"]]$set))
  dimnames(prob)[[1]] <- B$f[["1"]]$set
  aux <- rbeta(1, shape1 = 13, shape2 = 7)
  prob["0"] <- 1-aux; prob["1"] <- aux
  f <- function(x){
    if(x %in% B$f[["1"]]$set){
      return(unname(prob[as.character(x)]))}
    else{
```

```
81        return(0)}}
82      return(f)}

84  B$f[["2"]]$type <- "d"
85  B$f[["2"]]$set <- c(0,1)
86  B$f[["2"]][["1"]] <- function(y=
87                        rep(0,length(which(B$A[,"2"] == 1)))){
88    names(y) <- sortN(names(which(B$A[,"2"] == 1)))
89    f <- function(x){
90      if(x %in% B$f[["2"]]$set){
91        cat('\n Unknown, resort to ARA \n\n')}
92      else{
93        return(0)}}
94      return(f)}
95  B$f[["2"]][["2"]] <- function(y=
96                        rep(0,length(which(B$A[,"2"] == 1)))){
97    names(y) <- sortN(names(which(B$A[,"2"] == 1)))
98    f <- function(x){
99      if(x %in% B$f[["2"]]$set){
100       return(1)}
101     else{
102       return(0)}}
103     return(f)}

105 B$f[["3"]]$type <- "d"
106 B$f[["3"]]$set <- c(0,0.5,1)
107 B$f[["3"]][["1"]] <- function(y=
108                       rep(0,length(which(B$A[,"3"] == 1)))){
109   names(y) <- sortN(names(which(B$A[,"3"] == 1)))
110   prob <- array(0, dim = c(length(B$f[["3"]]$set),
111                            length(B$f[["1"]]$set),
112                            length(B$f[["2"]]$set)))
113   dimnames(prob)[[1]] <- B$f[["3"]]$set
114   dimnames(prob)[[2]] <- B$f[["1"]]$set
115   dimnames(prob)[[3]] <- B$f[["2"]]$set
116   prob[  "0","0","0"] <- 0.30; prob[  "0","1","0"] <- 0.40
117   prob["0.5","0","0"] <- 0.45; prob["0.5","1","0"] <- 0.40
118   prob[  "1","0","0"] <- 0.25; prob[  "1","1","0"] <- 0.20
119   prob[  "0","0","1"] <- 0.15; prob[  "0","1","1"] <- 0.25
120   prob["0.5","0","1"] <- 0.55; prob["0.5","1","1"] <- 0.50
121   prob[  "1","0","1"] <- 0.30; prob[  "1","1","1"] <- 0.25
122   if(y["5"] == 0){
123     f <- function(x){
124       if(x == 0){
125         return(1)}
126       else{
127         return(0)}}}
128   else{
129     f <- function(x){
130       if(x %in% B$f[["3"]]$set){
131         return(unname(prob[as.character(x),
132                            as.character(y["1"]),
133                            as.character(y["2"])]))}
```

```
134       else{
135         return(0)}}}
136       return(f)}
137 B$f[["3"]][["2"]] <- function(y=
138                       rep(0,length(which(B$A[,"3"] == 1)))){
139   names(y) <- sortN(names(which(B$A[,"3"] == 1)))
140   conc <- array(0, dim = c(length(B$f[["3"]]$set),
141                            length(B$f[["1"]]$set),
142                            length(B$f[["2"]]$set)))
143   dimnames(conc)[[1]] <- B$f[["3"]]$set
144   dimnames(conc)[[2]] <- B$f[["1"]]$set
145   dimnames(conc)[[3]] <- B$f[["2"]]$set
146   conc[  "0","0","0"] <-  69.70; conc[  "0","1","0"] <-  59.60
147   conc["0.5","0","0"] <- 104.55; conc["0.5","1","0"] <-  59.60
148   conc[  "1","0","0"] <-  58.08; conc[  "1","1","0"] <-  29.80
149   conc[  "0","0","1"] <-  84.85; conc[  "0","1","1"] <-  74.75
150   conc["0.5","0","1"] <- 311.12; conc["0.5","1","1"] <- 149.50
151   conc[  "1","0","1"] <- 169.70; conc[  "1","1","1"] <-  74.75
152   if(y["5"] == 0){
153     f <- function(x){
154       if(x == 0){
155         return(1)}
156       else{
157         return(0)}}}
158   else{
159     aux <- rdirichlet(1, conc[,as.character(y["1"]),
160                               as.character(y["2"])])
161     dimnames(aux)[[2]] <- dimnames(conc)[[1]]
162     f <- function(x){
163       if(x %in% B$f[["5"]]$set){
164         return(unname(aux[1,as.character(x)]))}
165       else{
166         return(0)}}}
167   return(f)}

169 B$f[["4"]]$type <- "d"
170 B$f[["4"]]$set <- c(0,1)
171 B$f[["4"]][["1"]] <- function(y=
172                       rep(0,length(which(B$A[,"4"] == 1)))){
173   names(y) <- sortN(names(which(B$A[,"4"] == 1)))
174   f <- function(x){
175     if(x %in% B$f[["4"]]$set){
176       return(1)}
177     else{
178       return(0)}}
179   return(f)}
180 B$f[["4"]][["2"]] <- function(y=
181                       rep(0,length(which(B$A[,"4"] == 1)))){
182   names(y) <- sortN(names(which(B$A[,"4"] == 1)))
183   conc <- array(0, dim = c(length(B$f[["4"]]$set),
184                            length(B$f[["3"]]$set)))
185   dimnames(conc)[[1]] <- B$f[["4"]]$set
186   dimnames(conc)[[2]] <- B$f[["3"]]$set
```

```
187   conc["0",  "0"] <-  1.00; conc["1",  "0"] <-   0.00
188   conc["0","0.5"] <- 39.40; conc["1","0.5"] <-  26.27
189   conc["0",  "1"] <- 69.70; conc["1",  "1"] <- 162.63
190   if(y["5"] == 0){
191     f <- function(x){
192       if(x == 0){
193         return(1)}
194       else{
195         return(0)}}}
196   else{
197     aux <- rdirichlet(1, conc[,as.character(y["3"])])
198     dimnames(aux)[[2]] <- dimnames(conc)[[1]]
199     f <- function(x){
200       if(x %in% B$f[["4"]]$set){
201         return(unname(aux[1,as.character(x)]))}
202       else{
203         return(0)}}}
204   return(f)}

206 B$f[["5"]]$type <- "d"
207 B$f[["5"]]$set <- c(0,1)
208 B$f[["5"]][["1"]] <- function(y=
209                       rep(0,length(which(B$A[,"5"] == 1)))){
210   names(y) <- sortN(names(which(B$A[,"5"] == 1)))
211   f <- function(x){
212     if(x %in% B$f[["5"]]$set){
213       cat('\n Unknown, resort to ARA \n\n')}
214     else{
215       return(0)}}
216   return(f)}
217 B$f[["5"]][["2"]] <- function(y=
218                       rep(0,length(which(B$A[,"5"] == 1)))){
219   names(y) <- sortN(names(which(B$A[,"5"] == 1)))
220   f <- function(x){
221     if(x %in% B$f[["5"]]$set){
222       return(1)}
223     else{
224       return(0)}}
225   return(f)}

227 B$f[["6"]]$type <- "d"
228 B$f[["6"]]$set <- c(0,0.25,0.5,0.75,1)
229 B$f[["6"]][["1"]] <- function(y=
230                       rep(0,length(which(B$A[,"6"] == 1)))){
231   names(y) <- sortN(names(which(B$A[,"6"] == 1)))
232   prob <- array(0, dim = c(length(B$f[["6"]]$set),
233                            length(B$f[["3"]]$set)))
234   dimnames(prob)[[1]] <- B$f[["6"]]$set
235   dimnames(prob)[[2]] <- B$f[["3"]]$set
236   prob[  "0",  "0"] <- 1.00
237   prob[  "0","0.5"] <- 0.20; prob["0.25","0.5"] <- 0.50
238   prob["0.5","0.5"] <- 0.30
239   prob[  "0",  "1"] <- 0.10; prob["0.25",  "1"] <- 0.15
```

```
240   prob["0.5",  "1"] <- 0.30; prob["0.75",  "1"] <- 0.25
241   prob[  "1",  "1"] <- 0.20
242   if(y["4"] == 0 | y["5"] == 0){
243     f <- function(x){
244       if(x == 0){
245         return(1)}
246       else{
247         return(0)}}}
248   else{
249     f <- function(x){
250       if(x %in% B$f[["6"]]$set){
251         return(unname(prob[as.character(x),
252                            as.character(y["3"])]))}
253       else{
254         return(0)}}}
255   return(f)}
256 B$f[["6"]][["2"]] <- function(y=
257                       rep(0,length(which(B$A[,"6"] == 1)))){
258   names(y) <- sortN(names(which(B$A[,"6"] == 1)))
259   conc <- array(0, dim = c(length(B$f[["6"]]$set),
260                            length(B$f[["3"]]$set)))
261   dimnames(conc)[[1]] <- B$f[["6"]]$set
262   dimnames(conc)[[2]] <- B$f[["3"]]$set
263   conc[  "0",  "0"] <-   1.00
264   conc[  "0","0.5"] <-  79.80; conc["0.25","0.5"] <- 199.50
265   conc["0.5","0.5"] <- 119.70
266   conc[  "0",  "1"] <-  89.90; conc["0.25",  "1"] <- 134.85
267   conc["0.5",  "1"] <- 269.70; conc["0.75",  "1"] <- 224.75
268   conc[  "1",  "1"] <- 179.80
269   if(y["4"] == 0 | y["5"] == 0){
270     f <- function(x){
271       if(x == 0){
272         return(1)}
273       else{
274         return(0)}}}
275   else{
276     aux <- rdirichlet(1, conc[,as.character(y["3"])])
277     dimnames(aux)[[2]] <- dimnames(conc)[[1]]
278     f <- function(x){
279       if(x %in% B$f[["6"]]$set){
280         return(unname(aux[1,as.character(x)]))}
281       else{
282         return(0)}}}
283   return(f)}

285 B$f[["7"]]$type <- "u"
286 B$f[["7"]][["1"]] <- function(){
287   m <- rep(0,length(which(B$A[,"7"] == 1)))
288   names(m) <- sortN(names(which(B$A[,"7"] == 1)))
289   m["1"] <-  -5; m["3"] <- -40; m["4"] <- -10; m["6"] <-  40
290   rho <- 0.06
291   c <- - unname(m["1"] + m["3"]*max(B$f[["3"]]$set) + m["4"])
292   f <- function(x){
```

```
293     names(x) <- sortN(names(which(B$A[,"7"] == 1)))
294     val <- unname(  m["1"]*x["1"] + m["3"]*x["3"]
295                   + m["4"]*x["4"] + m["6"]*x["6"])
296     return(1 - exp(-rho*(val + c)))}
297   return(f)}
298 B$f[["7"]][["2"]] <- function(){
299   cat('\n Not applicable \n\n')}

301 B$f[["8"]]$type <- "u"
302 B$f[["8"]][["1"]] <- function(){
303   cat('\n Not applicable \n\n')}
304 B$f[["8"]][["2"]] <- function(){
305   m <- rep(0,length(which(B$A[,"8"] == 1)))
306   names(m) <- sortN(names(which(B$A[,"8"] == 1)))
307   un <- runif(3, min = -0.5, max = 0.5)
308   m["2"] <- -1 + un[1]; m["3"] <-  35 + un[2]
309   m["5"] <- -5 + un[3]; m["6"] <- -35 - un[2]
310   rho <- runif(1, min = 0.05, max = 0.07)
311   c <- - max(0,unname(m["3"]*max(B$f[["3"]]$set) + m["5"]))
312   f <- function(x){
313     names(x) <- sortN(names(which(B$A[,"8"] == 1)))
314     val <- unname(  m["2"]*x["2"] + m["3"]*x["3"]
315                   + m["5"]*x["5"] + m["6"]*x["6"])
316     return(exp(rho*(val + c)))}
317   return(f)}

319 ## FUNCTIONS ###########################################################

321 # Sort array nodes:
322 #   Input -- x (array) - Nodes
323 #   Output -- ord (array) - Ordered nodes

325 sortN <- function(x){
326   return(as.character(sort(as.numeric(x))))}

328 # Agents' ID generator from BAID:
329 #   Input -- baid (object) - BAID
330 #            ag (int) - Agent
331 #   Output -- id (object) - Agent's ID

333 genID <- function(baid,ag){
334   id <- list()
335   id$n <- baid$n-1
336   id$V <- baid$V[ag,]
337   id$D <- sortN(baid$D[ag,]); id$D <- id$D[id$D != 0]
338   id$C <- sortN(c(baid$C,baid$D[-ag,])); id$C <- id$C[id$C != 0]
339   id$A <- baid$A[!rownames(baid$A) %in% baid$V[-ag,],
340                  !colnames(baid$A) %in% baid$V[-ag,]]
341   id$f <- baid$f[!rownames(baid$A) %in% baid$V[-ag,]]
342     for(i in names(id$f)){
343       id$f[[i]][["0"]] <- id$f[[i]][[as.character(ag)]]
344       id$f[[i]][["1"]] <- NULL
345       id$f[[i]][["2"]] <- NULL}
```

```
346   while(min(rowSums(id$A[!rownames(id$A) %in% id$V,])) == 0){
347     i <- names(which.min(rowSums(id$A[!rownames(id$A)
348                                       %in% id$V,])))
349     id$n <- id$n - 1
350     id$D <- id$D[id$D != i]
351     id$C <- id$C[id$C != i]
352     id$A <- id$A[!rownames(id$A) %in% i,!colnames(id$A) %in% i]
353     id$f[[i]] <- NULL
354     if(id$n == 1) break}
355   return(id)}

357 # Order graph nodes:
358 #   Input -- arcs (matrix) - Adjacency matrix
359 #   Output -- order (array) - Order of nodes (NULL if cycles)

361 orderN <- function(arcs){
362   w <- rownames(arcs)[rowSums(arcs) == 0]
363   if(length(w) != 0){
364     aux <- length(rownames(arcs)) - length(w)
365     if(aux > 1){
366       return(c(orderN(arcs[!rownames(arcs) %in% w,
367                            !colnames(arcs) %in% w]),w))}
368     else{
369       return(c(rownames(arcs)[!rownames(arcs) %in% w],w))}}
370   else if(length(which(!rownames(arcs) %in% w)) > 0){
371     cat('\n Graph is cyclic \n\n')
372     return(NULL)}
373   else{
374     return(w)}}

376 # Auxiliar function for computing SCCs:
377 #   Input -- v (char) - Visiting node
378 #            SCCs (object) - Auxiliar object to compute SCCs
379 #            arcs (matrix) - Adjacency matrix
380 #   Output -- SCCs (object) - Auxiliar object to compute SCCs

382 auxSCC <- function(v,SCCs,arcs){
383   SCCs$ind[1] <- SCCs$ind[1] + 1
384   SCCs$indN[1:2,v] <- SCCs$ind[1]
385   SCCs$S <- c(v,SCCs$S)
386   SCCs$stack[v] <- TRUE
387   for(w in colnames(arcs)[arcs[v,] != 0]){
388     if(SCCs$indN[1,w] == 0){
389       SCCs <- auxSCC(w,SCCs,arcs)
390       SCCs$indN[2,v] <- min(SCCs$indN[2,c(v,w)])}
391     else if(SCCs$stack[w]){
392       SCCs$indN[2,v] <- min(SCCs$indN[2,v],SCCs$indN[1,w])}}
393   if(SCCs$indN[1,v] == SCCs$indN[2,v]){
394     SCCs$ind[2] <- SCCs$ind[2] + 1
395     repeat{
396       w <- SCCs$S[1]
397       SCCs$S <- SCCs$S[SCCs$S != w]
398       SCCs$stack[w] <- FALSE
```

```
399       SCCs$indN[3,w] <- SCCs$ind[2]
400       if(w == v) break}}
401   return(SCCs)}

403 # Order graph SCCs:
404 #   Input -- arcs (matrix) - Adjacency matrix
405 #   Output -- order (array) - SCC for each node (num. ordered)

407 orderS <- function(arcs){
408   N <- rownames(arcs)
409   order <- rep(0,length(N)); names(order) <- N
410   SCCs <- list()
411   SCCs$ind <- rep(0,2)
412   SCCs$indN <- matrix(0,3,length(N)); colnames(SCCs$indN) <- N
413   SCCs$S <- NULL
414   SCCs$stack <- rep(FALSE,length(N)); names(SCCs$stack) <- N
415   for(i in N){
416     if(SCCs$indN[1,i] == 0){
417       SCCs <- auxSCC(i,SCCs,arcs)}}
418   order <- SCCs$indN[3,]
419   NS <- max(order)
420   arcsS <- matrix(0,NS,NS)
421   rownames(arcsS) <- c(1:max(order))
422   colnames(arcsS) <- c(1:max(order))
423   for(i in N){
424     for(j in colnames(arcs)[arcs[i,] != 0]){
425       if(order[i] != order[j]){
426         arcsS[order[i],order[j]] <- 1}}}
427   orderS <- orderN(arcsS)
428   ind <- 1
429   aux <- order
430   for(i in orderS){
431     order[aux == i] <- ind
432     ind <- ind + 1}
433   return(order)}

435 # Proper ID check:
436 #   Input -- id (object) - ID
437 #   Output -- proper (boolean) - TRUE if ID proper

439 propID <- function(id){
440   proper <- TRUE
441   # Orientation
442   if(length(id$V) != 1){
443     proper <- FALSE
444     cat('\n ID is not oriented \n\n')
445     return(proper)}
446   # Value node's successors
447   if(sum(id$A[id$V,]) != 0){
448     proper <- FALSE
449     cat('\n Value node with sucessors \n\n')
450     return(proper)}
451   # Cycles
```

```
452   w <- orderN(id$A)
453   if(length(w) != id$n){
454     proper <- FALSE
455     return(proper)}
456   # No-forgetting arcs
457   wD <- w[w %in% id$D]
458   for(i in wD){
459     wD <- wD[wD != i]
460     if(length(wD) != sum(id$A[i,wD])){
461       proper <- FALSE
462       cat('\n ID lacks some no-forgetting arcs \n\n')
463       return(proper)}}
464   return(proper)}

466 # Modified Bayes-Ball:
467 #   Input -- id (object) - ID
468 #            dec (array) - Decisions
469 #   Output -- relev (matrix) - Relevance graph for ID

471 bayesBall <- function(id,dec){
472   relev <- matrix(0,length(dec),length(id$D))
473   rownames(relev) <- dec; colnames(relev) <- id$D
474   wD <- orderN(id$A)
475   decD <- rev(wD[wD %in% id$D])
476   obs = NULL
477   for(i in decD){
478     J <- c(id$V,obs)
479     K <- c(i,rownames(id$A)[id$A[,i] == 1])
480     vis <- NULL; top <- NULL; bot <- NULL
481     child <- J; parent <- NULL
482     while(length(child) + length(parent) > 0){
483       if(length(child) > 0){
484         j <- child[1]
485         child <- child[child != j]
486         vis <- c(vis[vis != j],j)
487         if(!j %in% K){
488           if(!j %in% top){
489             top <- c(top,j)
490             for(k in rownames(id$A)){
491               if(id$A[k,j] == 1){
492                 child <- c(child[child != k],k)}}}
493           if(!j %in% bot){
494             bot <- c(bot,j)
495             for(k in colnames(id$A)){
496               if(id$A[j,k] == 1){
497                 parent <- c(parent[parent != k],k)}}}}}
498       else{
499         j <- parent[1]
500         parent <- parent[parent != j]
501         vis <- c(vis[vis != j],j)
502         if((j %in% K) & (!j %in% top)){
503           top <- c(top,j)
504           for(k in rownames(id$A)){
```

```
505             if(id$A[k,j] == 1){
506               child <- c(child[child != k],k)}}}
507         if((!j %in% K) & (!j %in% bot)){
508           bot <- c(bot,j)
509           for(k in colnames(id$A)){
510             if(id$A[k,j] == 1){
511               parent <- c(parent[parent != k],k)}}}}}
512     obs <- vis[vis %in% K]
513     for(j in dec){
514       if(j %in% top){
515         relev[j,i] <- 1}}}
516   return(relev)}

518 # Relevance graph generator from BAID:
519 #   Input -- baid (object) - BAID
520 #   Output -- relev (matrix) - Relevance graph (NULL not proper)

522 genRelev <- function(baid){
523   idD <- genID(baid,1)
524   idA <- genID(baid,2)
525   if(!propID(idD) | !propID(idA)) return(NULL)
526   dec <- c(idD$D,idA$D)
527   relevD <- bayesBall(idD,dec)
528   relevA <- bayesBall(idA,dec)
529   relev <- cbind(relevD,relevA)
530   return(relev)}

532 # Chance node removal:
533 #   Input -- id (object) - ID
534 #             v (char) - Chance node to be removed
535 #   Output -- idR (object) - Reduced ID and reduction

537 chance <- function(id,v){
538   idR <- list(); idR$id <- list(); idR$red <- list()
539   idR$id$n <- id$n - 1
540   idR$id$D <- id$D
541   idR$id$C <- id$C[!id$C %in% v]
542   idR$id$V <- id$V
543   idR$id$A <- id$A[!rownames(id$A) %in% v,
544                    !colnames(id$A) %in% v]
545   idR$id$A[names(which(id$A[,v] == 1)),id$V] <- 1
546   idR$id$f <- id$f[!names(id$f) %in% v]
547   idR$red$type <- "c"
548   idR$red$node <- v
549   idR$id$f[[idR$id$V]][["0"]] <- function(){
550     varO <- sortN(names(which(id$A[,id$V] == 1)))
551     varP <- sortN(names(which(id$A[,v] == 1)))
552     fV <- id$f[[id$V]][["0"]]()
553     f <- function(x){
554       names(x) <- sortN(names(which(idR$id$A[,idR$id$V] == 1)))
555       if(length(varP) > 0){
556         fv <- id$f[[v]][["0"]](x[varP])}
557       else{
```

```
558         fv <- id$f[[v]][["0"]]()}
559       faux <- function(z){
560         aux <- c(x[names(x) %in% var0],z)
561         names(aux)[length(aux)] <- v
562         aux <- aux[sortN(names(aux))]
563         return(fV(aux)*fv(z))}
564       val <- 0
565       if(id$f[[v]]$type == "d"){
566         for(z in id$f[[v]]$set){
567           val <- val + faux(z)}}
568       else{
569         for(z in seq(id$f[[v]]$set[1],id$f[[v]]$set[2],
570                      id$f[[v]]$set[3])){
571           val <- val + faux(z)}}
572       return(val)}
573     return(f)}
574   return(idR)}

576 # Decision node removal:
577 #   Input -- id (object) - ID
578 #            v (char) - Decision node to be removed
579 #   Output -- idR (object) - Reduced ID and reduction

581 decision <- function(id,v){
582   idR <- list(); idR$id <- list(); idR$red <- list()
583   idR$id$n <- id$n - 1
584   idR$id$D <- id$D[!id$D %in% v]
585   idR$id$C <- id$C
586   idR$id$V <- id$V
587   idR$id$A <- id$A[!rownames(id$A) %in% v,
588                    !colnames(id$A) %in% v]
589   idR$id$f <- id$f[!names(id$f) %in% v]
590   idR$red$type <- "d"
591   idR$red$node <- v
592   if(idR$id$n > 1){
593     aux <- function(id){
594       if(length(c(idR$id$C,idR$id$D)) > 1){
595         return(min(rowSums(
596         idR$id$A[!rownames(idR$id$A) %in% idR$id$V,])))}
597       else{
598         return(sum(
599         idR$id$A[!rownames(idR$id$A) %in% idR$id$V,]))}}
600     while(aux(idR$id) == 0){
601       if(length(c(idR$id$C,idR$id$D)) > 1){
602         i <- names(which.min(rowSums(
603              idR$id$A[!rownames(idR$id$A) %in% idR$id$V,])))}
604       else{
605         i <- c(idR$id$C,idR$id$D)}
606       idR$id$n <- idR$id$n - 1
607       idR$id$D <- idR$id$D[idR$id$D != i]
608       idR$id$C <- idR$id$C[idR$id$C != i]
609       idR$id$A <- idR$id$A[!rownames(idR$id$A) %in% i,
610                            !colnames(idR$id$A) %in% i]
```

```
611       idR$id$f[[i]] <- NULL
612       if(idR$id$n == 1) break}
613     idR$id$f[[idR$id$V]][["0"]] <- function(){
614       var0 <- sortN(names(which(id$A[,id$V] == 1)))
615       fV <- id$f[[id$V]][["0"]]()
616       f <- function(x){
617         names(x) <- sortN(names(which(
618                     idR$id$A[,idR$id$V] == 1)))
619         faux <- function(z){
620           aux <- c(x[names(x) %in% var0],z)
621           names(aux)[length(aux)] <- v
622           aux <- aux[sortN(names(aux))]
623           return(fV(aux))}
624         valD <- NULL
625         numD <- 1
626         if(id$f[[v]]$type == "d"){
627           for(z in id$f[[v]]$set){
628             valD <- c(valD,faux(z))
629             names(valD)[numD] <- z
630             numD <- numD + 1}}
631         else{
632           for(z in seq(id$f[[v]]$set[1],id$f[[v]]$set[2],
633                        id$f[[v]]$set[3])){
634             valD <- c(valD,faux(z))
635             names(valD)[numD] <- z
636             numD <- numD + 1}}
637         val <- max(valD)
638         return(val)}
639       return(f)}
640     idR$red$f <- function(){
641       var0 <- sortN(names(which(id$A[,id$V] == 1)))
642       fV <- id$f[[id$V]][["0"]]()
643       f <- function(x){
644         names(x) <- sortN(names(which(
645                     idR$id$A[,idR$id$V] == 1)))
646         val <- list()
647         faux <- function(z){
648           aux <- c(x[names(x) %in% var0],z)
649           names(aux)[length(aux)] <- v
650           aux <- aux[sortN(names(aux))]
651           return(fV(aux))}
652         valD <- NULL
653         numD <- 1
654         if(id$f[[v]]$type == "d"){
655           for(z in id$f[[v]]$set){
656             valD <- c(valD,faux(z))
657             names(valD)[numD] <- z
658             numD <- numD + 1}}
659         else{
660           for(z in seq(id$f[[v]]$set[1],id$f[[v]]$set[2],
661                        id$f[[v]]$set[3])){
662             valD <- c(valD,faux(z))
663             names(valD)[numD] <- z
```

```
664             numD <- numD + 1}}
665         val$eV <- max(valD)
666         val$dec <- as.numeric(names(which.max(valD)))
667         return(val)}
668       return(f)}}
669   else{
670     idR$id$f[[idR$id$V]][["0"]] <- function(){
671       fV <- id$f[[id$V]][["0"]]()
672       f <- function(){
673         valD <- NULL
674         numD <- 1
675         if(id$f[[v]]$type == "d"){
676           for(z in id$f[[v]]$set){
677             valD <- c(valD,fV(z))
678             names(valD)[numD] <- z
679             numD <- numD + 1}}
680         else{
681           for(z in seq(id$f[[v]]$set[1],id$f[[v]]$set[2],
682                        id$f[[v]]$set[3])){
683             valD <- c(valD,fV(z))
684             names(valD)[numD] <- z
685             numD <- numD + 1}}
686         val <- max(valD)
687         return(val)}
688       return(f)}
689     idR$red$f <- function(){
690       fV <- id$f[[id$V]][["0"]]()
691       f <- function(){
692         val <- list()
693         valD <- NULL
694         numD <- 1
695         if(id$f[[v]]$type == "d"){
696           for(z in id$f[[v]]$set){
697             valD <- c(valD,fV(z))
698             names(valD)[numD] <- z
699             numD <- numD + 1}}
700         else{
701           for(z in seq(id$f[[v]]$set[1],id$f[[v]]$set[2],
702                        id$f[[v]]$set[3])){
703             valD <- c(valD,fV(z))
704             names(valD)[numD] <- z
705             numD <- numD + 1}}
706         val$eV <- max(valD)
707         val$dec <- as.numeric(names(which.max(valD)))
708         return(val)}
709       return(f)}}
710   return(idR)}

712 # Arc inversion:
713 #   Input -- id (object) - ID
714 #            v (char) - Chance node to be inversed
715 #   Output -- inv (object) - Inversed ID
```

```
inverse <- function(id,v){
  inv <- id
  suc = orderN(id$A)[orderN(id$A)
                     %in% names(which(id$A[v,id$C] == 1))]
  for(i in suc){
    inv$A[v,i] <- 0
    inv$A[names(which(inv$A[,v] == 1)),i] <- 1
    inv$A[names(which(inv$A[,i] == 1)),v] <- 1
    inv$A[i,v] <- 1
    inv$f[[i]][["0"]] <- function(y=
                        rep(0,length(which(inv$A[,i] == 1)))){
      names(y) <- sortN(names(which(inv$A[,i] == 1)))
      var0 <- sortN(names(which(id$A[,i] == 1)))
      varP <- sortN(names(which(id$A[,v] == 1)))
      fv <- id$f[[v]][["0"]](y[varP])
      f <- function(x){
        faux <- function(z){
          aux <- c(y[names(y) %in% var0],z)
          names(aux)[length(aux)] <- v
          aux <- aux[sortN(names(aux))]
          fi <- id$f[[i]][["0"]](aux)
          return(id$f[[i]][["0"]](x)*fv(z))}
        val <- 0
        if(id$f[[v]]$type == "d"){
          for(z in id$f[[v]]$set){
            val <- val + faux(z)}}
        else{
          for(z in seq(id$f[[v]]$set[1],id$f[[v]]$set[2],
                       id$f[[v]]$set[3])){
            val <- val + faux(z)}}
        return(val)}
      return(f)}
    inv$f[[v]][["0"]] <- function(y=
                        rep(0,length(which(inv$A[,v] == 1)))){
      names(y) <- sortN(names(which(inv$A[,v] == 1)))
      var0 <- sortN(names(which(id$A[,i] == 1)))
      varN <- sortN(names(which(inv$A[,i] == 1)))
      varP <- sortN(names(which(id$A[,v] == 1)))
      f <- function(x){
        aux <- c(y[names(u) %in% var0],x)
        names(aux)[length(aux)] <- v
        aux <- aux[sortN(names(aux))]
        return(id$f[[i]][["0"]](aux)(y[i])*
               id$f[[v]][["0"]](y[varP])(x)/
               inv$f[[i]][["0"]](y[!names(y) %in% i])(y[i]))}
      return(f)}
    id <- inv}
  return(inv)}

# Shachter's procedure to evaluate IDs:
#   Input -- id (object) - ID
#   Output -- idR (object) - Reduced ID and reduction
```

```
shachter <- function(id){
  idR <- list()
  if(length(id$C) > 1){
    redC <- id$C[id$A[id$C,id$V] == 1 &
                   rowSums(id$A[id$C,]) == 1]}
  else{
    redC <- id$C[id$A[id$C,id$V] == 1 &
                   sum(id$A[id$C,]) == 1]}
  if(length(redC) != 0){
    idR <- chance(id,redC[1])}
  else {
    redD <- NULL
    for(i in id$D){
      if(id$A[i,id$V] == 1){
        if(all(names(which(id$A[,id$V] == 1))
               %in% c(names(which(id$A[,i] == 1)),i))){
          redD <- c(redD,i)}}}
    if(length(redD) != 0){
      idR <- decision(id,redD)}
    else{
      invC <- id$C[id$A[id$C,id$V] == 1 &
                     rowSums(id$A[id$C,c(id$D,id$V)]) == 1]
      id <- inverse(id,invC[1])
      idR <- chance(id,invC[1])}}
  return(idR)}

# Include adversarial decision in ID:
#   Input -- id (object) - ID
#            v (char) - Adversarial decision to be updated
#            rf (function) - Random decision function
#            K (num) - Monte Carlo iterations
#   Output -- idU (object) - Updated ID

advDec <- function(id,v,rf,K){
  idU <- id
  if(idU$f[[v]]$type == "c"){
    idU$f[[v]]$type <- "d"
    idU$f[[v]]$set <- seq(id$f[[v]]$set[1],id$f[[v]]$set[2],
                          id$f[[v]]$set[3])}
  if(length(which(id$A[,v] == 1)) == 0){
    prob <- rep(0,length(id$f[[v]]$set))
    names(prob) <- id$f[[v]]$set
    for(k in 1:K){
      faux <- rf()
      ind <- as.character(faux()$dec)
      prob[ind] <- prob[ind] + 1}
    prob <- prob/K
    idU$f[[v]][["0"]] <- function(){
      f <- function(x){
        return(unname(prob[as.character(x)]))}
      return(f)}}
  else{
    var <- sortN(names(which(id$A[,v] == 1)))
```

```
823     sets <- list()
824     dims <- rep(0,length(var))
825     names(dims) <- var
826     for(i in var){
827       if(id$f[[i]]$type == "d"){
828         sets[[i]] <- id$f[[i]]$set}
829       else{
830         sets[[i]] <- seq(id$f[[i]]$set[1],id$f[[i]]$set[2],
831                          id$f[[i]]$set[3])}
832       dims[i] <- length(sets[[i]])}
833     prob <- array(0, dim = c(length(id$f[[v]]$set),dims))
834     dimnames(prob)[[1]] <- id$f[[v]]$set
835     for(i in 2:(length(var)+1)){
836       dimnames(prob)[[i]] <- sets[[var[i-1]]]}
837     for(i in 1:prod(dims)){
838       y <- NULL
839       aux <- i
840       for(j in 1:length(var)){
841         r <- aux%%dims[j]
842         if(r > 0){
843           y <- c(y,sets[[var[j]]][r])
844           aux <- aux%/%dims[j] + 1}
845         else{
846           y <- c(y,sets[[var[j]]][dims[j]])
847           aux <- aux%/%dims[j]}}
848       for(k in 1:K){
849         faux <- rf()
850         ind1 <- which(id$f[[v]]$set == faux(y)$dec)
851         ind <- ind1 + (i-1)*length(id$f[[v]]$set)
852         prob[ind] <- prob[ind] + 1}}
853     prob <- prob/K
854     idU$f[[v]][["0"]] <- function(y){
855       aux <- 1
856       for(i in length(var):1){
857         ind2 <- which.min(abs(sets[[var[i]]] - y[i]))
858         aux <- unname((aux - 1)*dims[var[i]] + ind2)}
859       f <- function(x){
860         ind1 <- which(id$f[[v]]$set == x)
861         ind <- ind1 + (aux-1)*length(id$f[[v]]$set)
862         return(prob[ind])}
863       return(f)}}
864   return(idU)}

866 # Acyclic case computation algorithm:
867 #   Input -- baid (object) - BAID (acyclic relevance graph)
868 #            ordSCC (array) - SCC for each node (num. ordered)
869 #            K (num) - Monte Carlo iterations
870 #   Output -- value (object) - Expected value and decisions

872 acAlg <- function(baid,ordSCC,K){
873   idD <- genID(baid,1)
874   idA <- genID(baid,2)
875   value <- list(); value$D <- list(); value$A <- list()
```

```
876   for(i in 1:length(ordSCC)){
877     v <- names(which(ordSCC == i))
878     if(v %in% baid$D[1,]){
879       while(v %in% idD$D){
880         idR <- shachter(idD)
881         idD <- idR$id}
882       value$D[[idR$red$node]] <- idR$red$f()}
883     else{
884       while(v %in% idA$D){
885         idR <- shachter(idA)
886         idA <- idR$id}
887       idD <- advDec(idD,idR$red$node,idR$red$f,K)
888       value$A[[idR$red$node]] <- idD$f[[idR$red$node]][["0"]]}}
889   return(value)}

891 # Cyclic case computation algorithm:
892 #   Input -- baid (object) - BAID (cyclic relevance graph)
893 #            ordSCC (array) - SCC for each node (num. ordered)
894 #            K (num) - Monte Carlo iterations
895 #   Output -- value (object) - Expected value and decisions

897 cAlg <- function(baid,ordSCC,K){
898   idD <- genID(baid,1)
899   idA <- genID(baid,2)
900   ordN <- rev(orderN(baid$A))
901   value <- list(); value$D <- list(); value$A <- list()
902   for(i in 1:max(ordSCC)){
903     nod <- ordN[ordN %in% names(which(ordSCC == i))]
904     for(v in nod[nod %in% idA$D]){
905       while(v %in% idA$D){
906         idR <- shachter(idA)
907         idA <- idR$id}
908       idD <- advDec(idD,idR$red$node,idR$red$f,K)
909       value$A[[idR$red$node]] <- idD$f[[idR$red$node]][["0"]]}
910     for(v in nod[nod %in% idD$D]){
911       while(v %in% idD$D){
912         idR <- shachter(idD)
913         idD <- idR$id}
914       value$D[[idR$red$node]] <- idR$red$f()}}
915   return(value)}

917 ## COMPUTATION ##################################################

919 # Start time:

921 sTime <- Sys.time()

923 # Seed for random variables:

925 set.seed(12345)

927 # Determine relevance graph:
```

```
R <- genRelev(B)

# Solve with appropriate algorithm:

if(length(R) == 0){
  stop("BAID is not proper")
}else{
  ordSCC <- orderS(R)
  if(length(ordSCC) == max(ordSCC)){
    cat('\n Relevance graph is acyclic \n\n')
    value <- acAlg(B,ordSCC,K)}
  else{
    cat('\n Relevance graph is cyclic \n\n')
    value <- cAlg(B,ordSCC,K)}}

# Computational time:

cTime <- Sys.time() - sTime

## OUTPUT ######################################################

# Defender's optimal decisions and expected values:

cat('\n Optimal decisions:',
    '\n\n Decision D1  -> ',
    value$D[["1"]]()$dec, '/ Exp. Val =',
    format(value$D[["1"]]()$eV,digits=3,nsmall=3),
    '\n\n Decision D2 (D1 = 0, S1 =   0, A2 = 0)  -> ',
    value$D[["4"]](c(0,0,0))$dec, '/ Exp. Val =',
    format(value$D[["4"]](c(0,0,0))$eV,digits=3,nsmall=3),
    '\n Decision D2 (D1 = 0, S1 =   0, A2 = 1)  -> ',
    value$D[["4"]](c(0,0,1))$dec, '/ Exp. Val =',
    format(value$D[["4"]](c(0,0,1))$eV,digits=3,nsmall=3),
    '\n Decision D2 (D1 = 0, S1 = 0.5, A2 = 1)  -> ',
    value$D[["4"]](c(0,0.5,1))$dec, '/ Exp. Val =',
    format(value$D[["4"]](c(0,0.5,1))$eV,digits=3,nsmall=3),
    '\n Decision D2 (D1 = 0, S1 =   1, A2 = 1)  -> ',
    value$D[["4"]](c(0,1,1))$dec, '/ Exp. Val =',
    format(value$D[["4"]](c(0,1,1))$eV,digits=3,nsmall=3),
    '\n Decision D2 (D1 = 1, S1 =   0, A2 = 0)  -> ',
    value$D[["4"]](c(1,0,0))$dec, '/ Exp. Val =',
    format(value$D[["4"]](c(1,0,0))$eV,digits=3,nsmall=3),
    '\n Decision D2 (D1 = 1, S1 =   0, A2 = 1)  -> ',
    value$D[["4"]](c(1,0,1))$dec, '/ Exp. Val =',
    format(value$D[["4"]](c(1,0,1))$eV,digits=3,nsmall=3),
    '\n Decision D2 (D1 = 1, S1 = 0.5, A2 = 1)  -> ',
    value$D[["4"]](c(1,0.5,1))$dec, '/ Exp. Val =',
    format(value$D[["4"]](c(1,0.5,1))$eV,digits=3,nsmall=3),
    '\n Decision D2 (D1 = 1, S1 =   1, A2 = 1)  -> ',
    value$D[["4"]](c(1,1,1))$dec, '/ Exp. Val =',
    format(value$D[["4"]](c(1,1,1))$eV,digits=3,nsmall=3),
    '\n\n')
```

```
# Attack probabilities:

cat('\n Attack probabilities (0 / 1):',
    '\n\n Action A1  -> ',
    format(value$A[["2"]]()(0),digits=3,nsmall=3), '/',
    format(value$A[["2"]]()(1),digits=3,nsmall=3),
    '\n\n Action A2 (D1 = 0, A1 = 0)  -> ',
    format(value$A[["5"]](c(0,0))(0),digits=3,nsmall=3), '/',
    format(value$A[["5"]](c(0,0))(1),digits=3,nsmall=3),
    '\n Action A2 (D1 = 0, A1 = 1)  -> ',
    format(value$A[["5"]](c(0,1))(0),digits=3,nsmall=3), '/',
    format(value$A[["5"]](c(0,1))(1),digits=3,nsmall=3),
    '\n Action A2 (D1 = 1, A1 = 0)  -> ',
    format(value$A[["5"]](c(1,0))(0),digits=3,nsmall=3), '/',
    format(value$A[["5"]](c(1,0))(1),digits=3,nsmall=3),
    '\n Action A2 (D1 = 1, A1 = 1)  -> ',
    format(value$A[["5"]](c(1,1))(0),digits=3,nsmall=3), '/',
    format(value$A[["5"]](c(1,1))(1),digits=3,nsmall=3),
    '\n\n')

# Computational time:

cat('\n Time difference of', format(cTime,digits=3,nsmall=3),
    '\n\n')
```

B.2 Concept uncertainty (MATLAB)

```
%% SETTINGS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Monte Carlo iterations:

K = 10^4;

% Average data-stream rate's integration bounds:

boundT = [-10^3 10^3];

% Observations' integration bounds:

boundX = [-10^3 10^3];

%% INPUT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Attacker's choices:

a = [-25 0];

% Defender's assessment of attacker's parameters [lower upper]:
%   mu - Average data-stream bitrate
%   sigma - Standard deviation data-stream rate
%   rho - Standard deviation observations
```

```
25 %   d - Defender's decision range
26 %   alpha - Effort cost
27 %   beta - Misestimation benefit

29 mu = [175 225];
30 sigma = [20 30];
31 rho = [5 15];
32 d = [-10 10];
33 alpha = [3 5];
34 beta = [-0.3 -0.1];

36 param = [mu; sigma; rho; d; alpha; beta];

38 %% COMPUTATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

40 % Start time:

42 tic

44 % Seed for random variables:

46 rng(12345)

48 % Attack probabilities simulation (Bayesian adversary):

50 prob = simBay(a,param,K,boundT,boundX);

52 % Computational time:

54 cTime = toc;

56 %% OUTPUT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

58 % Attack probabilities (Bayesian adversary):

60 fprintf('\n')
61 disp('Attack probabilities (Bayesian adversary):')
62 fprintf('\n')
63 disp(['Action ', num2str(prob(1,1),'%.0f'), ...
64       '  ->  ', num2str(prob(1,3)/100,'%.3f')])
65 disp(['Action   ', num2str(prob(2,1),'%.0f'), ...
66       '  ->  ', num2str(prob(2,3)/100,'%.3f')])

68 % Computational time:

70 fprintf('\n')
71 disp(['Time difference of ', num2str(cTime,'%.3f'), ' secs'])
72 fprintf('\n')

74 %% FUNCTIONS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

76 % Attack probabilities simulation (Bayesian adversary):
77 %   Input -- a (array) - Attacker's choices
```

```
%             param (matrix) - Attacker's parameters
%             K (num) - Monte Carlo iterations
%             boundT (array) - Theta integration bounds
%             boundX (array) - X integration bounds
%   Output -- prob (matrix) - Attack probabilities

function prob = simBay(a,param,K,boundT,boundX)
  prob = zeros(1,K);
  for i = 1:K
    mu = unifrnd(param(1,1),param(1,2));
    sigma = unifrnd(param(2,1),param(2,2));
    rho = unifrnd(param(3,1),param(3,2));
    dl = param(4,1); du = param(4,2);
    alpha = unifrnd(param(5,1),param(5,2));
    beta = unifrnd(param(6,1),param(6,2));
    prob(i) = a(1);
    val1 = val(a(1));
    for j = 2:length(a)
      val2 = val(a(j));
      if(val2 < val1)
        prob(i) = a(j);
        val1 = val2;
      end
    end
  end
  prob = tabulate(prob);
  function w = val(y)
    f = @auxInt;
    w = integral2(f,boundX(1),boundX(2),boundT(1),boundT(2));
    function z = auxInt(x,t)
      z1 = 1/(rho*sqrt(2*pi))*exp(-(x-(t+y)).^2/(2*rho^2));
      z2 = 1/(sigma*sqrt(2*pi))*exp(-(t-mu).^2/(2*sigma^2));
      z3 =  alpha*(du-dl)*abs(y) ...
          + beta/3*((x+du-t).^3-(x+dl-t).^3);
      z = z1.*z2.*z3;
    end
  end
end
```

B.3 Adversarial hypothesis testing (R)

```
## SETTINGS ##############################################################

# Required libraries:

library(gtools)
library(rootSolve)

# Monte Carlo iterations:

K <- 10^5
```

```
12 # Observation's integration upper bound:

14 boundX <- 500

16 ## INPUT ###########################################################

18 # Hypotheses:

20 theta <- c(2,1)

22 # Non-adversarial attack probabilities:

24 probNoadv <- c(1/2,1/6,1/3)
25 names(probNoadv) <- c("a0","a1","a2")

27 ## FUNCTIONS ########################################################

29 # Attack probabilities simulation (AHT problem):
30 # Input -- theta (array) - Hypotheses
31 #          K (num) - Monte Carlo iterations
32 #          boundX (num) - X integration upper bound
33 # Output -- prob (array) - Attack probabilities

35 simAHT <- function(theta,K,boundX){
36   prob <- rep(0,3)
37   names(prob) <- c("a0","a1","a2")
38   piA <- rep(0,2)
39   sigma <- rep(0,2)
40   alpha <- rep(0,2)
41   beta <- rep(0,2)
42   f1 <- function(x){
43     dgamma(x, shape = alpha[1], rate = beta[1])}
44   f2 <- function(x){
45     dgamma(x, shape = alpha[2], rate = beta[2])}
46   eps <- rep(0,3)
47   g <- function(y){
48     1/(1 + 2*
49     (eps[1]*exp(-2*y) + eps[2]*exp(-y) + eps[3]*exp(-4*y))/
50     (eps[1]*exp(-y) + eps[2]*exp(-y/2) + eps[3]*exp(-2*y)))}
51   cA <- 0
52   cost1 <- function(x){
53     piA[1]*(1-g(x))*f1(x) + cA*piA[2]*g(x)*f2(x)}
54   cost2 <- function(x){
55     piA[1]*(1-g(2*x))*f1(x) + cA*piA[2]*g(2*x)*f2(x)}
56   cost3 <- function(x){
57     piA[1]*(1-g(x/2))*f1(x) + cA*piA[2]*g(x/2)*f2(x)}
58   psiA <- rep(0,3)
59   jOpt <- 0
60   for(k in 1:K){
61     piA[2] <- runif(1, min = 1/4, max = 3/4)
62     piA[1] <- 1 - piA[2]
63     sigma <- runif(2, min = 1/2, max = 2)
```

```
64      alpha <- theta^2/sigma
65      beta <- theta/sigma
66      eps <- rdirichlet(1, alpha = c(1,1,1))
67      cA <- runif(1, min = 1/2, max = 1)
68      psiA[1] <- try(integrate(cost1, lower = 0,
69                      upper = boundX)$value, silent = TRUE)
70      psiA[2] <- try(integrate(cost2, lower = 0,
71                      upper = boundX)$value, silent = TRUE)
72      psiA[3] <- try(integrate(cost3, lower = 0,
73                      upper = boundX)$value, silent = TRUE)
74      jOpt <- suppressWarnings(which.min(psiA))
75      prob[jOpt] <- prob[jOpt] + 1}
76    return(prob/K)}

78  # Defender's condition for d0 (AHT problem):
79  #   Input -- y (num) - Observed data
80  #            theta (array) - Hypotheses
81  #            prob (array) - Attack probabilities
82  #   Output -- cond (num) - Evaluated condition (d0 if <=0)

84  condAHT <- function(y,theta,prob){
85    cond = (  1/2*(  prob[1]*exp(-theta[2]*y)
86                   + prob[2]*exp(-theta[2]*y/2)
87                   + prob[3]*exp(-theta[2]*2*y))
88            - 3/4*(  prob[1]*exp(-theta[1]*y)
89                   + prob[2]*exp(-theta[1]*y/2)
90                   + prob[3]*exp(-theta[1]*2*y)))
91    return(cond)}

93  ## COMPUTATION ##################################################

95  # Start time:

97  sTime <- Sys.time()

99  # Seed for random variables:

101 set.seed(12345)

103 # Non-adversarial defender's optimal condition:

105 obsNoadv <- uniroot.all(
106   function(y){condAHT(y,theta,probNoadv)},c(0,boundX))

108 # Adversarial defender's optimal condition:

110 probAdv <- simAHT(theta,K,boundX)
111 obsAdv <- uniroot.all(
112   function(y){condAHT(y,theta,probAdv)},c(0,boundX))

114 # Computational time:

116 cTime <- Sys.time() - sTime
```

```
## OUTPUT ########################################################

# Non-adversarial defender's optimal condition:

cat('\n Non-adversarial defender optimal condition:',
    '\n\n Choose d0 if observation less or equal than',
    format(obsNoadv,digits=3,nsmall=3),
    '\n\n')

# Non-adversarial attack probabilities:

cat('\n Non-adversarial attack probabilities:',
    '\n\n Action a0  -> ',
    format(probNoadv["a0"],digits=3,nsmall=3),
    '\n Action a1  -> ',
    format(probNoadv["a1"],digits=3,nsmall=3),
    '\n Action a2  -> ',
    format(probNoadv["a2"],digits=3,nsmall=3),
    '\n\n')

# Adversarial defender's optimal condition:

cat('\n Adversarial defender optimal condition:',
    '\n\n Choose d0 if observation less or equal than',
    format(obsAdv,digits=3,nsmall=3),
    '\n\n')

# Adversarial attack probabilities:

cat('\n Adversarial attack probabilities:',
    '\n\n Action a0  -> ',
    format(probAdv["a0"],digits=3,nsmall=3),
    '\n Action a1  -> ',
    format(probAdv["a1"],digits=3,nsmall=3),
    '\n Action a2  -> ',
    format(probAdv["a2"],digits=3,nsmall=3),
    '\n\n')

# Computational time:

cat('\n Time difference of', format(cTime,digits=3,nsmall=3),
    '\n\n')
```

B.4 Batch acceptance model (R)

```
## SETTINGS ######################################################

# Required libraries:

library(gtools)
```

```
# Maximum number of injected items:

Y1 <- 4

# Maximum observed batch size:

N <- 8

# Monte Carlo iterations:

K <- 10^3

# Dirichlet process iterations:

DP <- 500

# Average reception rate's integration upper bound:

boundG <- 500

## INPUT #########################################################

# Batch acceptante parameters:

batch <- list()

# Defender's parameters:
#   gamma - Reception rate parameters (Gamma distribution)
#   theta - Acceptability parameters (Beta distribution)
#   c - Expected opportunity costs

batch$theta <- c(9,1)
batch$gamma <- c(5,1)
batch$c <- 0.90

# Defender's assessment of attacker's parameters:
#   f1 - Expected A-fault injection cost
#   f2 - Expected A-fault modification cost
#   g - Expected A-fault gain
#   h - Expected O-fault gain
#   rho - Acceptability conc. parameter (Dirichlet process)

batch$f1 <- c(0.25,0.50)
batch$f2 <- c(0.30,0.60)
batch$g  <- c(0.80,1.00)
batch$h  <- c(0.00,0.25)
batch$rho <- 100

## FUNCTIONS #####################################################

# Attacker's distribution over defences given batch size:
#   Input -- n (num) - Batch size
```

```
#            batch (list) - Defender's assumptions
#   Output -- pD0 (num) - Simulated acceptance probability

simD0 <- function(n,batch){
  bound <- rep(1,2)
  if(n > 0){
    avg <- 1
    for(k in 0:(n-1)){
      avg <- avg*(batch$theta[1]+k)/(sum(batch$theta)+k)}
    aux <- batch$theta[2]/(2*(batch$theta[1]+n)+batch$theta[2])
    bound[1] <- avg*(1-aux)/2
    bound[2] <- avg*(1+aux)/2}
  pD0 <- runif(1, min = bound[1], max = bound[2])
  return(pD0)}

# Theta's Dirichlet process:
#   Input -- batch (list) - Defender's assumptions
#            DP (num) - Dirichlet process iterations
#   Output -- thetaP (array) - Theta (names) & prob (value)

thetaDP <- function(batch,DP){
  theta <- rbeta(1, shape1 = batch$theta[1],
                    shape2 = batch$theta[2])
  thetaP <- table(theta)
  pr <- NULL
  aux <- 0
  for(i in 2:DP){
    pr <- thetaP/(batch$rho+1)
    if(length(thetaP) > 1){
      for(j in 2:length(thetaP)){
        pr[j] <- pr[j] + pr[j-1]}}
    aux <- runif(1, min = 0, max = 1)
    if(aux > pr[length(pr)]){
      theta <- c(theta,
                 rbeta(1, shape1 = batch$theta[1],
                          shape2 = batch$theta[2]))}
    else{
      theta <- c(theta,as.numeric(names(which.min(
                 pr[pr >= aux])))))}
    thetaP <- table(theta)}
  return(thetaP/DP)}

# Attack probabilities simulation (batch acceptance):
#   Input -- m (num) - Batch size
#            batch (list) - Defender's assumptions
#            Y1 (num) - Maximum number of injected items
#            K (num) - Monte Carlo iterations
#            DP (num) - Dirichlet process iterations
#   Output -- prob (matrix) - Attack probabilities
#                             (1st col = y2=0 / 2nd col = y2=m)

simBatch <- function(m,batch,Y1,K,DP){
  prob <- matrix(0,Y1+1,2)
```

```
112   rownames(prob) <- c(0:Y1); colnames(prob) <- c(0,m)
113   f1 <- 0
114   f2 <- 0
115   g <- 0
116   h <- 0
117   thetaP <- NULL
118   pD0 <- rep(0,Y1+1)
119   psiA <- matrix(0,Y1+1,m+1)
120   yOpt <- 0
121   for(k in 1:K){
122     h <- runif(1, min = batch$h[1], max = batch$h[2])
123     g <- runif(1, min = batch$g[1], max = batch$g[2])
124     f1 <- runif(1, min = batch$f1[1], max = batch$f1[2])
125     f2 <- runif(1, min = batch$f2[1], max = batch$f2[2])
126     if(m > 0){
127       thetaP <- thetaDP(batch,DP)
128       intTheta <- 0
129       for(t in names(thetaP)){
130         sumX = 0
131         for(x in 0:m){
132           sumX <-  sumX + (m-x)*dbinom(x, size = m,
133                                        prob = as.numeric(t))}
134         intTheta <- intTheta + sumX*unname(thetaP[t])}
135       for(y1 in 0:Y1){
136         pD0[y1+1] <- simD0(m+y1,batch)
137         for(y2 in 0:1){
138           psiA[y1+1,y2+1] <- (  y1*(f1-g*pD0[y1+1])
139                               + y2*(f2-g*pD0[y1+1])*m
140                               - h*(1-y2)*pD0[y1+1]*intTheta)}}}
141     else{
142       for(y1 in 0:Y1){
143         pD0[y1+1] <- simD0(m+y1,batch)
144         psiA[y1+1,1] <- y1*(f1-g*pD0[y1+1])}}
145     yOpt <- which.min(psiA)
146     prob[yOpt] <- prob[yOpt] + 1}
147   return(prob/K)}

149 # Defender's batch acceptance:
150 #   Input -- batch (list) -- Defender's assumptions
151 #            Y1 (num) - Maximum number of injected items
152 #            N (num) - Maximum observed batch size
153 #            K (num) - Monte Carlo iterations
154 #            DP (num) - Dirichlet process iterations
155 #            boundG (num) - Gamma integration upper bound
156 #   Output -- accept (object) - Acceptance per size and info.

158 acceptBatch <- function(batch,Y1,N,K,DP,boundG){
159   accept <- list()
160   accept$cond <- array(0,N+1,list(0:N))
161   accept$val <- array(0,N+1,list(0:N))
162   accept$prob <- list()
163   cond <- array(0,N+1,list(0:N))
164   prob <- matrix(0,N+1,N+1)
```

```
165   prob0 <- rep(0,N+1)
166   aux <- NULL
167   for(i in 0:N){
168     aux <- simBatch(i,batch,Y1,K,DP)
169     accept$prob[[as.character(i)]] <- aux
170     for(j in i:min(i+Y1,N)){
171       prob[i+1,j+1] <- sum(aux[j-i+1,])}
172     prob0[i+1] <- aux["0","0"]}
173   ETheta <- rep(1,N+1)
174   if(N > 0){
175     for(i in 0:(N-1)){
176       ETheta[i+2] <- ETheta[i+1]*(batch$theta[1]+i)/
177                      (sum(batch$theta)+i)}}
178   EGamma <- rep(0,N+1)
179   for(i in 0:N){
180     pGamma <- function(g){
181       sum = 0
182       for(j in 0:i){
183         sum = sum + dpois(j, lambda = g)*prob[j+1,i+1]}
184       pG <- dgamma(g, shape = batch$gamma[1],
185                       rate = batch$gamma[2])
186       return(dpois(i, lambda = g)*pG/sum)}
187     EGamma[i+1] <- try(integrate(pGamma, lower = 0,
188                         upper = M)$value,silent = TRUE)}
189   EGamma <- as.numeric(EGamma)
190   for(i in 0:N){
191     if(prob0[i+1] > 0){
192       cond[i+1] <- (  1/(1+batch$c)
193                     - EGamma[i+1]*ETheta[i+1]*prob0[i+1])}
194     else{
195       cond[i+1] <- 1/(1+batch$c)}}
196   accept$val <- 1/(1+batch$c) - cond
197   accept$cond <- cond <= 0
198   return(accept)}

200 ## COMPUTATION ##################################################

202 # Start time:

204 sTime <- Sys.time()

206 # Seed for random variables:

208 set.seed(12345)

210 # Defender's batch acceptance:

212 accept <- acceptBatch(batch,Y1,N,K,DP,boundG)

214 # End and computing time:

216 cTime <- Sys.time() - sTime
```

```
## OUTPUT ##########################################################

# Defender's optimal acceptance per size:

cat('\n Defender optimal acceptance:',
    '\n\n Size 0  -> ', accept$cond["0"],
    '\n Size 1  -> ', accept$cond["1"],
    '\n Size 2  -> ', accept$cond["2"],
    '\n Size 3  -> ', accept$cond["3"],
    '\n Size 4  -> ', accept$cond["4"],
    '\n Size 5  -> ', accept$cond["5"],
    '\n Size 6  -> ', accept$cond["6"],
    '\n Size 7  -> ', accept$cond["7"],
    '\n Size 8  -> ', accept$cond["8"],
    '\n\n')

# Defender's optimal acceptance values:

cat('\n Defender optimal acceptance values (q3):',
    '\n\n Size 0  -> ',
    format(accept$val["0"],digits=3,nsmall=3),
    '\n Size 1  -> ', format(accept$val["1"],digits=3,nsmall=3),
    '\n Size 2  -> ', format(accept$val["2"],digits=3,nsmall=3),
    '\n Size 3  -> ', format(accept$val["3"],digits=3,nsmall=3),
    '\n Size 4  -> ', format(accept$val["4"],digits=3,nsmall=3),
    '\n Size 5  -> ', format(accept$val["5"],digits=3,nsmall=3),
    '\n Size 6  -> ', format(accept$val["6"],digits=3,nsmall=3),
    '\n Size 7  -> ', format(accept$val["7"],digits=3,nsmall=3),
    '\n Size 8  -> ', format(accept$val["8"],digits=3,nsmall=3),
    '\n\n')

# Attack probabilities:

cat('\n Attack probabilities:',
    '\n\n Size 0 \n [y2 = 0] y1 = 0 / y1 = 1 / y1 = 2 -> ',
    format(accept$prob[["0"]][1,1],digits=3,nsmall=3), '/',
    format(accept$prob[["0"]][2,1],digits=3,nsmall=3), '/',
    format(accept$prob[["0"]][3,1],digits=3,nsmall=3),
    '\n          y1 = 3 / y1 = 4          -> ',
    format(accept$prob[["0"]][4,1],digits=3,nsmall=3), '/',
    format(accept$prob[["0"]][5,1],digits=3,nsmall=3),
    '\n\n Size 1 \n [y2 = 0] y1 = 0 / y1 = 1 / y1 = 2 -> ',
    format(accept$prob[["1"]][1,1],digits=3,nsmall=3), '/',
    format(accept$prob[["1"]][2,1],digits=3,nsmall=3), '/',
    format(accept$prob[["1"]][3,1],digits=3,nsmall=3),
    '\n          y1 = 3 / y1 = 4          -> ',
    format(accept$prob[["1"]][4,1],digits=3,nsmall=3), '/',
    format(accept$prob[["1"]][5,1],digits=3,nsmall=3),
    '\n [y2 = 1] y1 = 0 / y1 = 1 / y1 = 2 -> ',
    format(accept$prob[["1"]][1,2],digits=3,nsmall=3), '/',
    format(accept$prob[["1"]][2,2],digits=3,nsmall=3), '/',
    format(accept$prob[["1"]][3,2],digits=3,nsmall=3),
    '\n          y1 = 3 / y1 = 4          -> ',
```

```
271     format(accept$prob[["1"]][4,2],digits=3,nsmall=3), '/',
272     format(accept$prob[["1"]][5,2],digits=3,nsmall=3),
273     '\n\n Size 2 \n [y2 = 0] y1 = 0 / y1 = 1 / y1 = 2 -> ',
274     format(accept$prob[["2"]][1,1],digits=3,nsmall=3), '/',
275     format(accept$prob[["2"]][2,1],digits=3,nsmall=3), '/',
276     format(accept$prob[["2"]][3,1],digits=3,nsmall=3),
277     '\n           y1 = 3 / y1 = 4           -> ',
278     format(accept$prob[["2"]][4,1],digits=3,nsmall=3), '/',
279     format(accept$prob[["2"]][5,1],digits=3,nsmall=3),
280     '\n [y2 = 2] y1 = 0 / y1 = 1 / y1 = 2 -> ',
281     format(accept$prob[["2"]][1,2],digits=3,nsmall=3), '/',
282     format(accept$prob[["2"]][2,2],digits=3,nsmall=3), '/',
283     format(accept$prob[["2"]][3,2],digits=3,nsmall=3),
284     '\n           y1 = 3 / y1 = 4           -> ',
285     format(accept$prob[["2"]][4,2],digits=3,nsmall=3), '/',
286     format(accept$prob[["2"]][5,2],digits=3,nsmall=3),
287     '\n\n Size 3 \n [y2 = 0] y1 = 0 / y1 = 1 / y1 = 2 -> ',
288     format(accept$prob[["3"]][1,1],digits=3,nsmall=3), '/',
289     format(accept$prob[["3"]][2,1],digits=3,nsmall=3), '/',
290     format(accept$prob[["3"]][3,1],digits=3,nsmall=3),
291     '\n           y1 = 3 / y1 = 4           -> ',
292     format(accept$prob[["3"]][4,1],digits=3,nsmall=3), '/',
293     format(accept$prob[["3"]][5,1],digits=3,nsmall=3),
294     '\n [y2 = 3] y1 = 0 / y1 = 1 / y1 = 2 -> ',
295     format(accept$prob[["3"]][1,2],digits=3,nsmall=3), '/',
296     format(accept$prob[["3"]][2,2],digits=3,nsmall=3), '/',
297     format(accept$prob[["3"]][3,2],digits=3,nsmall=3),
298     '\n           y1 = 3 / y1 = 4           -> ',
299     format(accept$prob[["3"]][4,2],digits=3,nsmall=3), '/',
300     format(accept$prob[["3"]][5,2],digits=3,nsmall=3),
301     '\n\n Size 4 \n [y2 = 0] y1 = 0 / y1 = 1 / y1 = 2 -> ',
302     format(accept$prob[["4"]][1,1],digits=3,nsmall=3), '/',
303     format(accept$prob[["4"]][2,1],digits=3,nsmall=3), '/',
304     format(accept$prob[["4"]][3,1],digits=3,nsmall=3),
305     '\n           y1 = 3 / y1 = 4           -> ',
306     format(accept$prob[["4"]][4,1],digits=3,nsmall=3), '/',
307     format(accept$prob[["4"]][5,1],digits=3,nsmall=3),
308     '\n [y2 = 4] y1 = 0 / y1 = 1 / y1 = 2 -> ',
309     format(accept$prob[["4"]][1,2],digits=3,nsmall=3), '/',
310     format(accept$prob[["4"]][2,2],digits=3,nsmall=3), '/',
311     format(accept$prob[["4"]][3,2],digits=3,nsmall=3),
312     '\n           y1 = 3 / y1 = 4           -> ',
313     format(accept$prob[["4"]][4,2],digits=3,nsmall=3), '/',
314     format(accept$prob[["4"]][5,2],digits=3,nsmall=3),
315     '\n\n Size 5 \n [y2 = 0] y1 = 0 / y1 = 1 / y1 = 2 -> ',
316     format(accept$prob[["5"]][1,1],digits=3,nsmall=3), '/',
317     format(accept$prob[["5"]][2,1],digits=3,nsmall=3), '/',
318     format(accept$prob[["5"]][3,1],digits=3,nsmall=3),
319     '\n           y1 = 3 / y1 = 4           -> ',
320     format(accept$prob[["5"]][4,1],digits=3,nsmall=3), '/',
321     format(accept$prob[["5"]][5,1],digits=3,nsmall=3),
322     '\n [y2 = 5] y1 = 0 / y1 = 1 / y1 = 2 -> ',
323     format(accept$prob[["5"]][1,2],digits=3,nsmall=3), '/',
```

```
324     format(accept$prob[["5"]][2,2],digits=3,nsmall=3), '/',
325     format(accept$prob[["5"]][3,2],digits=3,nsmall=3),
326     '\n           y1 = 3 / y1 = 4           -> ',
327     format(accept$prob[["5"]][4,2],digits=3,nsmall=3), '/',
328     format(accept$prob[["5"]][5,2],digits=3,nsmall=3),
329     '\n\n Size 6 \n [y2 = 0] y1 = 0 / y1 = 1 / y1 = 2 -> ',
330     format(accept$prob[["6"]][1,1],digits=3,nsmall=3), '/',
331     format(accept$prob[["6"]][2,1],digits=3,nsmall=3), '/',
332     format(accept$prob[["6"]][3,1],digits=3,nsmall=3),
333     '\n           y1 = 3 / y1 = 4           -> ',
334     format(accept$prob[["6"]][4,1],digits=3,nsmall=3), '/',
335     format(accept$prob[["6"]][5,1],digits=3,nsmall=3),
336     '\n [y2 = 6] y1 = 0 / y1 = 1 / y1 = 2 -> ',
337     format(accept$prob[["6"]][1,2],digits=3,nsmall=3), '/',
338     format(accept$prob[["6"]][2,2],digits=3,nsmall=3), '/',
339     format(accept$prob[["6"]][3,2],digits=3,nsmall=3),
340     '\n           y1 = 3 / y1 = 4           -> ',
341     format(accept$prob[["6"]][4,2],digits=3,nsmall=3), '/',
342     format(accept$prob[["6"]][5,2],digits=3,nsmall=3),
343     '\n\n Size 7 \n [y2 = 0] y1 = 0 / y1 = 1 / y1 = 2 -> ',
344     format(accept$prob[["7"]][1,1],digits=3,nsmall=3), '/',
345     format(accept$prob[["7"]][2,1],digits=3,nsmall=3), '/',
346     format(accept$prob[["7"]][3,1],digits=3,nsmall=3),
347     '\n           y1 = 3 / y1 = 4           -> ',
348     format(accept$prob[["7"]][4,1],digits=3,nsmall=3), '/',
349     format(accept$prob[["7"]][5,1],digits=3,nsmall=3),
350     '\n [y2 = 7] y1 = 0 / y1 = 1 / y1 = 2 -> ',
351     format(accept$prob[["7"]][1,2],digits=3,nsmall=3), '/',
352     format(accept$prob[["7"]][2,2],digits=3,nsmall=3), '/',
353     format(accept$prob[["7"]][3,2],digits=3,nsmall=3),
354     '\n           y1 = 3 / y1 = 4           -> ',
355     format(accept$prob[["7"]][4,2],digits=3,nsmall=3), '/',
356     format(accept$prob[["7"]][5,2],digits=3,nsmall=3),
357     '\n\n Size 8 \n [y2 = 0] y1 = 0 / y1 = 1 / y1 = 2 -> ',
358     format(accept$prob[["8"]][1,1],digits=3,nsmall=3), '/',
359     format(accept$prob[["8"]][2,1],digits=3,nsmall=3), '/',
360     format(accept$prob[["8"]][3,1],digits=3,nsmall=3),
361     '\n           y1 = 3 / y1 = 4           -> ',
362     format(accept$prob[["8"]][4,1],digits=3,nsmall=3), '/',
363     format(accept$prob[["8"]][5,1],digits=3,nsmall=3),
364     '\n [y2 = 8] y1 = 0 / y1 = 1 / y1 = 2 -> ',
365     format(accept$prob[["8"]][1,2],digits=3,nsmall=3), '/',
366     format(accept$prob[["8"]][2,2],digits=3,nsmall=3), '/',
367     format(accept$prob[["8"]][3,2],digits=3,nsmall=3),
368     '\n           y1 = 3 / y1 = 4           -> ',
369     format(accept$prob[["8"]][4,2],digits=3,nsmall=3), '/',
370     format(accept$prob[["8"]][5,2],digits=3,nsmall=3),
371     '\n\n')

373 # Computational time:

375 cat('\n Time difference of', format(cTime,digits=3,nsmall=3),
376     '\n\n')
```

Notation

Abbreviations

AHT Adversarial Hypothesis Testing

ARA Adversarial Risk Analysis

ASDT Adversarial Statistical Decision Theory

BAID Bi-Agent Influence Diagram

CIP Critical Infrastructure Protection

DG Differential Game

EW Electronic Warfare

ID Influence Diagram

MAID Multi-Agent Influence Diagram

PRI Pulse Repetition Interval

SCC Strongly Connected Component

SDG Stochastic Differential Game

SDT Statistical Decision Theory

Seq. A-D Sequential Attack-Defend

Seq. D-A Sequential Defend-Attack

Seq. D-A with PI Sequential Defend-Attack with Private Information

Seq. D-A-D Sequential Defend-Attack-Defend

Sim. D-A Simultaneous Defend-Attack

(BA)IDs

Conditional arrow

Informational arrow


Defender's decision node


Defender's chance node


Defender's chance node relative to an attacker's choice


Defender's utility node


Attacker's decision node


Attacker's chance node


Attacker's chance node relative to a defender's choice


Attacker's utility node


Shared chance node


SCC


Selected decision node


Selected chance node


Selected chance node relative to an attacker's choice


Selected chance node relative to a defender's choice


Selected SCC

References

Banks, D.L.; Ríos, J. & Ríos Insua, D. (2015). *Adversarial Risk Analysis*, 2016 ed. CRC Press, Boca Raton, FL.

Bao, S.; Zhang, C.; Ouyang, M. & Miao, L. (2017). An integrated tri-level model for enhancing the resilience of facilities against intentional attacks. *Annals of Operations Research*, `https://doi.org/10.1007/s10479-017-2705-y`.

Barni, M. & Pérez-González, F. (2013). Coping with the enemy: Advances in adversary-aware signal processing. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8682–8686.

Barni, M. & Tondi, B. (2014). Binary hypothesis testing game with training data. *IEEE Transactions on Information Theory*, 60(8) 4848–4866.

Basar, T. & Li, S. (1989). Distributed computation of Nash equilibria in linear-quadratic stochastic differential games. *SIAM Journal on Control and Optimization*, 27(3) 563–578.

Bedford, T. & Cooke, R.M. (2001). *Probabilistic Risk Analysis: Foundations and Methods*, 2003 ed. Cambridge University Press, Cambridge, UK.

Bensoussan, A.; Kantarcioglu, M. & Hoe, S. (2010). A game-theoretical approach for finding optimal strategies in a botnet defense model. In *Decision and Game Theory for Security: GameSec 2010*, 2010 ed., Springer, Berlin, Germany, 135–148.

Berger, J.O. (2003). Could Fisher, Jeffreys and Neyman have agreed on testing? *Statistical Science*, 18(1) 1–32.

Berger, J.O. & Sellke, T. (1987). Testing a point null hypothesis: The irreconcilability of p values and evidence. *Journal of the American Statistical Association*, 87(397) 112–122.

Bhattacharjya, D. & Shachter, R.D. (2012). Formulating asymmetric decision problems as decision circuits. *Decision Analysis*, 9(2) 138–145.

Bielza, C.; Müller, P. & Ríos Insua, D. (1999). Decision analysis by augmented probability simulation. *Management Science*, 45(7) 995–1007.

Bielza, C. & Shenoy, P.P. (1999). A comparison of graphical techniques for asymmetric decision problems. *Management Science*, 45(11) 1552–1569.

Boulet, P.; Darte, A.; Silber, G.A. & Vivien, F. (1998). Loop parallelization algorithms: From parallelism extraction to code generation. *Parallel Computing*, 24(3-4) 421–444.

Brown, G.G.; Carlyle, W.M.; Salmerón, J. & Wood, R.K. (2006). Defending critical infrastructure. *Interfaces*, 36(6) 530–544.

Brown, G.G.; Carlyle, W.M. & Wood, R.K. (2008). Optimizing department of Homeland Security defense investments: Applying Defender-Attacker(-Defender) optimization to terror risk assessment and mitigation. Appears as Appendix E of *Department of Homeland Security Bioterrorist Risk Assessment: A Call for Change*, National Academies Press, Washington, DC.

Call, H.J. & Miller, W.A. (1990). A comparison of approaches and implementations for automating decision analysis. *Reliability Engineering and System Safety*, 30(1-3) 115–162.

Camerer, C.F.; Ho, T.H. & Chong, J.K. (2004). A cognitive hierarchy model of games. *The Quarterly Journal of Economics*, 119(3) 861–898.

Chung, K.L. (1968). *A Course in Probability Theory*, 2001 ed. Academic Press, San Diego, CA.

Clyde, M. & George, E.I. (2004). Model uncertainty. *Statistical Science*, 19(1) 81–94.

Conlisk, J. (1996). Why bounded rationality? *Journal of Economic Literature*, 24(2) 669–700.

Cooke, R.M. (1991). *Experts in Uncertainty: Opinion and Subjective Probability in Science*, 1991 ed. Oxford University Press, New York, NY.

Cox Jr., L.A. (2009). Game theory and risk analysis. *Risk Analysis*, 29(8) 1062–1068.

Dalvi, N.; Domingos, P.; Mausam; Sanghai, S. & Verma, D. (2004). Adversarial classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 99–108.

Darwin, C.R. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, 1869 ed. John Murray, London, UK.

Demirer, R. & Shenoy, P.P. (2006). Sequential valuation networks for asymmetric decision problems. *European Journal of Operational Research*, 169(1) 286–309.

Dyer, J.S. & Sarin, R.K. (1982). Relative risk aversion. *Management Science*, 28(8) 875–886.

French, S. & Ríos Insua, D. (2000). *Kendall's Library of Statistics 9: Statistical Decision Theory*, 2000 ed. Wiley, New York, NY.

Friedman, A. (1972). Stochastic differential games. *Journal of Differential Equations*, 11(1) 79–108.

GeNIe (1998). BayesFusion, LLC. `GeNIe` modeler: Complete modeling freedom. Pittsburgh, PA, `https://www.bayesfusion.com/genie-modeler`.

Gibbons, R. (1992). *Game Theory for Applied Economists*, 1992 ed. Princeton University Press, Princeton, NJ.

Gigerenzer, G. & Selten, R. (2002). *Bounded Rationality: The Adaptive Toolbox*, 2002 ed. MIT Press, Cambridge, MA.

Gómez-Corral, A.; Ríos Insua, D.; Ruggeri, F. & Wiper, M. (2015). Bayesian inference of Markov processes. In *Wiley StatsRef: Statistics Reference Online*, `https://doi.org/10.1002/9781118445112.stat07837`.

Gómez Esteban, P.; Liu, A.; Ríos Insua, D. & González-Ortega, J. (exp. 2019). Competition and cooperation in a community of autonomous agents. Submitted to *Autonomous Robots*.

González-Ortega, J.; Radovic, V. & Ríos Insua, D. (2018). Utility elicitation. In *Elicitation: The Science and Art of Structuring Judgement*, 2018 ed., Springer, Cham, Switzerland, 241–264.

González-Ortega, J.; Ríos Insua, D. & Cano, J. (2018). Adversarial risk analysis for bi-agent influence diagrams: An algorithmic approach. *European Journal of Operational Research*, in press, `https://doi.org/10.1016/j.ejor.2018.09.015`.

González-Ortega, J.; Ríos Insua, D.; Ruggeri, F. & Soyer, R. (exp. 2019). Hypothesis testing in presence of adversaries. Submitted to *The American Statistician*.

Hargreaves-Heap, S.P. & Varoufakis, Y. (1995). *Game Theory: A Critical Introduction*, 2004 ed. Routledge, New York, NY.

Harsanyi, J.C. (1967). Games with incomplete information played by "Bayesian" players, I-III. Part I. The basic model. *Management Science*, 14(3) 159–182.

Harsanyi, J.C. (1982). Subjective probability and the theory of games: Comments on Kadane and Larkey's Paper. *Management Science*, 28(2) 120–124.

Hausken, K. (2011). Strategic defense and attack of series systems when agents move sequentially. *IIE Transactions*, 43(7) 483–504.

Hausken, K. & Bier, V.M. (2011). Defending against multiple different attackers. *European Journal of Operational Research*, 211(2) 370–384.

Hearing, B. & Franklin, J. (Mar. 1, 2016). *Drone Detection and Classification Methods and Apparatus*. U.S. Patent No.: US9275645B2.

Hock, M. & Soyer, R. (2006). A Bayesian approach to signal analysis of pulse trains. In *Bayesian Monitoring, Control and Optimization*, 2007 ed., CRC Press, Boca Raton, FL, 215–243.

Hoeting, J.A.; Madigan, D.; Raftery, A.E. & Volinsky, C.T. (1999). Bayesian model averaging: A tutorial. *Statistical Science*, 14(4) 382–417.

Howard, R.A. & Matheson, J.E. (2005). Influence diagrams. *Decision Analysis*, 2(3) 127–143.

Isaacs, R. (1965). *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*, 1999 ed. Dover Publications, Mineola, NY.

Johnson, V.E. (2013). Uniformly most powerful Bayesian tests. *Annals of Statistics*, 41(4) 1716–1741.

Kadane, J.B. (2009). Bayesian thought in early modern detective stories: Monsieur Lecoq, C. Auguste Dupin and Sherlock Holmes. *Statistical Science*, 24(2) 238–243.

Kadane, J.B. & Larkey, P.D. (1982). Subjective probability and the theory of games. *Management Science*, 28(2) 113–120.

Keeney, G.L. & von Winterfeldt, D. (2010). Identifying and structuring the objectives of terrorists. *Risk Analysis*, 30(12) 1803–1816.

Keeney, R.L. (2007). Modeling values for anti-terrorism analysis. *Risk Analysis*, 27(3) 585–596.

Koller, D. & Milch, B. (2003). Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1) 181–221.

La Mura, P. (2000). Game networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 335–342.

Lehmann, E.L. & Casella, G. (1983). *Theory of Point Estimation*, 1998 ed. Springer, New York, NY.

Luce, R.D. & Raiffa, H. (1957). *Games and Decisions: Introduction and Critical Survey*, 2012 ed. Dover Publications, Mineola, NY.

MATLAB (2017b). The MathWorks Inc. `MATLAB`: The language of technical computing. Natick, MA, `http://www.mathworks.com`.

Merrick, J. & Parnell, G.S. (2011). A comparative analysis of PRA and intelligent adversary methods for counterterrorism risk management. *Risk Analysis*, 31(9) 1488–1510.

Miller III, A.C.; Merkhofer, M.W.; Howard, R.A.; Matheson, J.E. & Rice, T.R. (1976). *Development of Automated Aids for Decision Analysis*, technical report. Stanford Research Institute, Menlo Park, CA.

Myerson, R.B. (1991). *Game Theory: Analysis of Conflict*, 1997 ed. Harvard University Press, Cambridge, MA.

Nash, J.F. (1951). Non-cooperative games. *Annals of Mathematics*, 54(2) 286–295.

Nassi, R.; Ben-Netanel, R.; Shamir, A. & Elovici, Y. (2018). Game of drones – Detecting streamed POI from encrypted FPV channel. Preprint, `https://arxiv.org/abs/1801.03074`.

Naveiro, R.; Redondo, A.; Ríos Insua, D. & Ruggeri, F. (2018). Adversarial classification: An adversarial risk analysis approach. Preprint, `https://arxiv.org/abs/1802.07513`.

Nisio, M. (1981). *Stochastic Control Theory: Dynamic Programming Principle*, 2015 ed. Springer, Tokyo, Japan.

O'Hagan, A.; Buck, C.E.; Daneshkhah, A.; Eiser, J.R.; Garthwaite, P.H.; Jenkinson, D.J.; Oakley, J. E. & Rakow, T. (2006). *Uncertain Judgements: Eliciting Experts' Probabilities*, 2006 ed. Wiley, Chichester, UK.

Parnell, G.S.; Banks, D.L.; Borio, L.; Brown, G.G; Cox Jr., L.A.; Gannon, J.; Harvill, E.T.; Kunreuther, H.C.; Morse, S.S.; Pappaioanou, M.; Pollock, S.M.; Singpurwalla, N.D. & Wilson, A.G. (2007). *Interim Report on Methodological Improvements to the Department of Homeland Security's Biological Agent Risk Analysis*, technical report. National Academies Press, Washington, DC.

Parry, G.W. (1996). The characterization of uncertainty in probabilistic risk assessments of complex systems. *Reliability Engineering & System Safety*, 54(2-3) 119–126.

Paté-Cornell, E. & Guikema, S. (2002). Probabilistic modeling of terrorist threats: A systems analysis approach to setting priorities among countermeasures. *Military Operations Research*, 7(4) 5–23.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, 2014 ed. Morgan Kaufmann, San Francisco, CA.

Pontryagin, L.S. (1961). *Mathematical Theory of Optimal Processes*, 2018 ed. Routledge, London, UK.

R (2008). `R` Development Core Team: A language and environment for statistical computing. Vienna, Austria, `http://www.R-project.org`.

Raiffa, H. (1982). *The Art and Science of Negotiation*, 2003 ed. Harvard University Press, Cambridge, MA.

Raiffa, H.; Richardson, J. & Metcalfe, D. (2002). *Negotiation Analysis: The Science and Art of Collaborative Decision Making*, 2002 ed. Harvard University Press, Cambridge, MA.

Refsgaard, J.C.; van der Sluijs, J.P.; Højberg, A.L. & Vanrolleghem, P.A. (2007). Uncertainty in the environmental modelling process – A framework and guidance. *Environmental Modelling & Software*, 22(11) 1543–1556.

Ríos, J. & Ríos Insua, D. (2012). Adversarial risk analysis for counterterrorism modeling. *Risk Analysis*, 32(5) 894–915.

Ríos Insua, D.; Banks, D.L. & Ríos, J. (2016). Modeling opponents in adversarial risk analysis. *Risk Analysis*, 36(4) 742–755.

Ríos Insua, D.; Banks, D.L.; Ríos, J. & González-Ortega, J. (2017). Adversarial risk analysis. In *Wiley StatsRef: Statistics Reference Online*, `https://doi.org/10.1002/9781118445112.stat07972`.

Ríos Insua, D.; Banks, D.L.; Ríos, J. & González-Ortega, J. (exp. 2019). Structured expert judgement modelling through adversarial risk analysis. Forthcoming.

Ríos Insua, D.; Cano, J.; Shim, W.; Massacci, F. & Schmitz, A. (2013). *SECONOMICS "Socio-Economics meets Security". Deliverable 5.1. Basic models for security risk analysis*, technical report. European Union.

Ríos Insua, D.; González-Ortega, J.; Banks, D.L. & Ríos, J. (2018). Concept uncertainty in adversarial statistical decision theory. In *The Mathematics of the Uncertain: A Tribute to Pedro Gil*, 2018 ed., Springer, Cham, Switzerland, 527–542.

Ríos Insua, D.; Ríos, J. & Banks, D.L. (2009). Adversarial risk analysis. *Journal of the American Statistical Association*, 104(486) 841–854.

Ríos Insua, D.; Ruggeri, F.; Alfaro, C. & Gomez, J. (2016). Robustness for adversarial risk analysis. In *Robustness Analysis in Decision Aiding, Optimization, and Analytics*, 2016 ed., Springer, Cham, Switzerland, 39–58.

Ríos Insua, D.; Ruggeri, F. & M. Wiper, M. (2012). *Bayesian Analysis of Stochastic Process Models*, 2012 ed. Wiley, Chichester, UK.

Savage, L.J. (1954). *The Foundations of Statistics*, 1972 ed. Dover Publications, New York, NY.

Sevillano, J.C.; Ríos Insua, D. & Ríos, J. (2012). Adversarial risk analysis: The Somali pirates case. *Decision Analysis*, 9(2) 86–95.

Shachter, R.D. (1986). Evaluating influence diagrams. *Operations Research*, 34 (6) 871–882.

Shachter, R.D. (1998). Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 480–487.

Shin, D.H.; Jung, D.H.; Kim, D.C.; Ham, J.W. & Park, S.O. (2017). A distributed FMCW radar system based on fiber-optic links for small drone detection. *IEEE Transactions on Instrumentation and Measurement*, 66(2) 340–347.

Singpurwalla, N.D.; Arnold, B.C.; Gastwirth, J.L.; Gordon, A.S. & Ng, H.K.T. (2016). Adversarial and amiable inference in medical diagnosis, reliability and survival analysis. *International Statistical Review*, 84(3) 390–412.

Smith, J.Q. (1996). Plausible Bayesian games. In *Proceedings of the Fifth Valencia International Meeting*, 387–406.

Stahl, D.O. & Wilson, P.W. (1995). On players' models of other players: Theory and experimental evidence. *Games and Economic Behavior*, 10(1) 218–254.

Tetlock, P.E. & Gardner, D. (2015). *Superforecasting: The Art and Science of Prediction*, 2015 ed. Broadway Books, New York, NY.

Thwaites, P.A. & Smith, J.Q. (2017). A graphical method for simplifying Bayesian games. *Reliability Engineering & System Safety*, 179(1) 3–11.

Trundle, S.S. & Slavin, A.J. (Mar. 30, 2017). *Drone Detection Systems.* U.S. Patent Publication No.: US20170092138A1.

Tygar, J.D. (2011). Adversarial machine learning. *IEEE Internet Computing*, 15 (5) 4–6.

von Neumann, J. & Morgenstern, O. (1944). *Theory of Games and Economic Behavior (Sixtieth-Anniversary Edition)*, 2007 ed. Princeton University Press, Princeton, NJ.

Wald, A. (1950). Statistical decision functions. In *Breakthroughs in Statistics Volume I: Foundations and Basic Theory*, 1992 ed., Springer, New York, NY, 342–357.

Walker, W.E.; Harremoës, P.; Rotmans, J.; van der Sluijs, J.P.; van Asselt, M.B.A.; Janssen, P. & Krayer von Krauss, M.P. (2003). Defining uncertainty: A conceptual basis for uncertainty management in model-based decision support. *Integrated Assessment*, 4(1): 5–17.

Wang, S. & Banks, D.L. (2011). Network routing for insurgency: An adversarial risk analysis framework. *Naval Research Logistics*, 58(6) 595–607.

Wasserstein, R.L. & Lazar, N.A. (2016). The ASA's statement on p-values: Context, process, and purpose. *The American Statistician*, 70(2) 129–133.

Zhang, C. & Ramirez-Marquez, J.E. (2013). Protecting critical infrastructures against intentional attacks: A two-stage game with incomplete information. *IIE Transactions*, 45(3) 244–258.

Zhang, C.; Ramirez-Marquez, J.E. & Wang, J. (2015). Critical infrastructure protection using secrecy – A discrete simultaneous game. *European Journal of Operational Research*, 242(1) 212–221.

Zhang, H. & Zenios, S. (2008). A dynamic principal-agent model with hidden information: Sequential optimality through truthful state revelation. *Operations Research*, 56(3) 681–696.

Zhuang, J. & Bier, V.M. (2007). Balancing terrorism and natural disasters: Defensive strategy with endogenous attacker effort. *Operations Research*, 55(5) 976–991.

Zhuang, J. & Bier, V.M. (2010). Reasons for secrecy and deception in Homeland-Security resource allocation. *Risk Analysis*, 30(12) 1737–1743.

Zhuang, J.; Bier, V.M. & Alagoz, O. (2010). Modeling secrecy and deception in a multiple-period attacker-defender signaling game. *European Journal of Operational Research*, 203(2) 409–418.